Forum SixtyEight 68

AUG. 1983

The Motorola User's Monthly Forum Vol. 1 No. 2

Build performance into your system

with OS-9** software tools

Unix*-based, multitasking, modular, and versatile: these key features are some of the reasons why more 6809 computer manufacturers have selected OS-9 as their standard operating system than any other. And OS-9 has been put to work by thousands of users in almost every conceivable computer application in business, science, industry, education, and government.

Your operating system should not be a barrier between you and your computer. OS-9 is very friendly and easy to use. Its modular structure makes it easy to customize, plus its comprehensive documentation shows you exactly how to interface it to just about any I/O device.

OS-9's advanced features unleash the performance potential of almost any 6809 computer — large or small. In many respects the OS-9/6809 combination is more powerful than many *minicomputers!*

There are two basic versions of OS-9. Both have the same basic features and capabilities. OS-9 Level One runs on small to medium sized systems having up to 64K memory. The Level Two version runs on medium to large size systems having memory management hardware and up to 1 megabyte of memory, and includes record and file locking for multiuser database applications.

Here are just a few reasons why you should insist on OS-9 for your microcomputer system.

- **Over 40 utility commands**
- **Friendly "Shell" command interpreter**
- Tree-structured multilevel file directories
- Full timesharing support with log-in and file security
- Fast, secure random and sequential access files
- Comprehensive English language error messages
- Compact real-time multitasking executive
- Hardware or software memory management
- Device independent interrupt-driven I/O
- Fully ROMable for small control systems
- Standard versions available from manufacturers of most popular 6809 computers

OS-9 PASCAL Language Compiler

- most complete and versatile **PASCAL available for the 6809**
- **capable of generating P-code for interpretive execution while debugging OR**
- **highly optimized 6809 assembly language source code output for maximum speed**
- "virtual memory" P-code **interpreter lets you run large PASCAL programs**

CIS COBOL * Compiler**

- **ideal for most demanding business applications**
- features ISAM, Debug, ACCEPT/DISPLAY and Interprogram Communications modules
- **retains full compatibility with CP/M software**
- **meets ANSI 1974 Level One COBOL standard and is GSA certified**
- Also available-FORMS 2 automatic program generator for easy interative design of screen oriented applications.

BASIC09 Structured Basic Interactive Compiler**

- fastest and most comprehensive full Basic language available for the 6809
- combines standard Basic with the best features of PASCAL
- features compiler speed, interpreter friendliness and superlative debugging facilities
- option available: Run B...a **ROMable run-time system for compiled Basic 09**

C Language Compiler

- **complete implementation of the UNIX version 7 C language**
- **includes INT, CHAR, SIGNED, UNSIGNED, FLOAT AND LONG data types, structures, unions, standard C library and a full preprocessor with macro definitions**
- **generates fully reentrant 6809 assembly language source code output**

For more information contact your computer supplier, or

MICROWARE

Microware Systems Corporation 5835 Grand Avenue, Des Moines, Iowa 50312 515-279-8844 • Telex 910-520-2535

*Unix is a trademark of Bell Laboratories. ***CIS Cobol is a trademark of Micro Focus, Inc. OS-9** and Basic09 are trademarks of Microware and Motorola, Inc.

The GMX 6809 CPU III

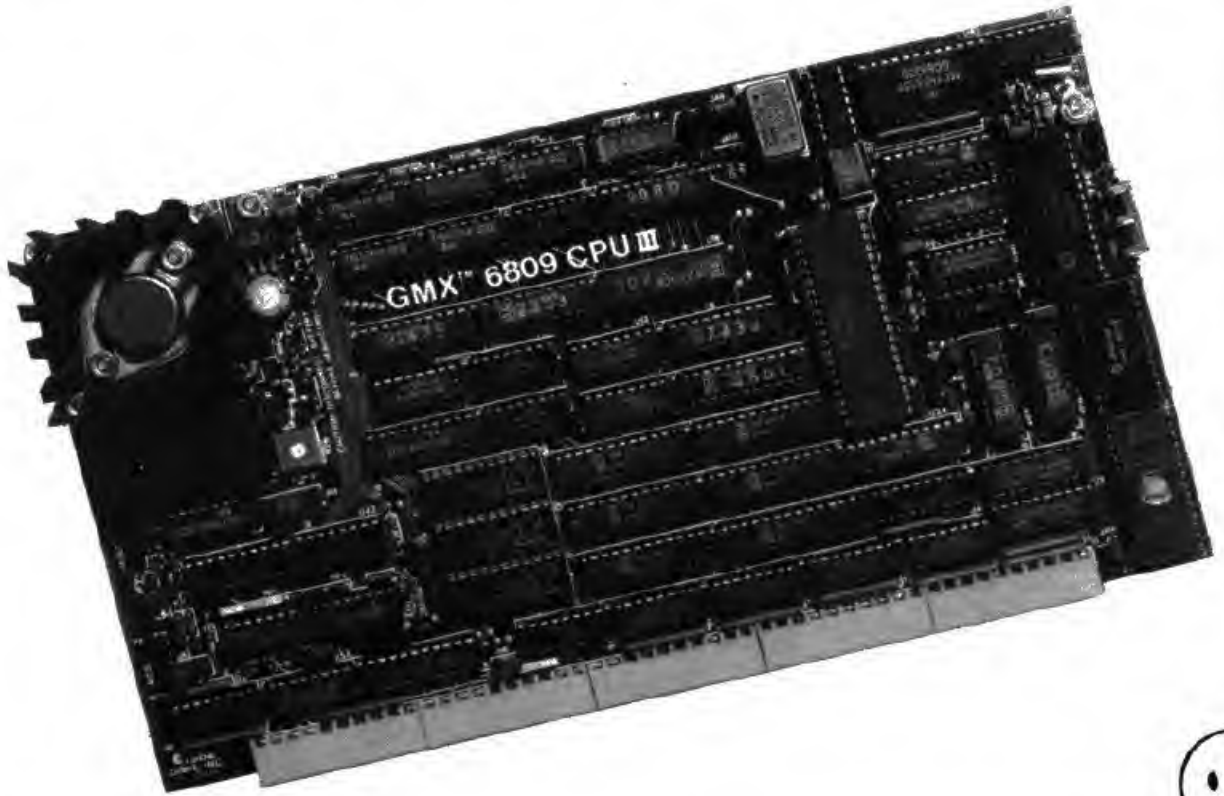
The GIMIX 6809 CPU III board is an advanced design, specifically intended for use with multi-user, multi-tasking operating systems.

Built on a multi-layer circuit board and utilizing high-speed, high-density logic, the GMX CPU III enhances the performance of the 2MHz. 68B09 by providing such features as 1 byte/microsecond DMA block transfers from memory to memory or between memory and I/O devices, and advanced memory management with 2K segments and segment attributes. The board automatically arbitrates DMA contention between the on board DMA and external DMA devices such as disk controllers. The 2K memory segments allow more efficient memory usage. The segment attributes allow the trapping of out-of-range memory references, write protection, and a hardware single step function for software debugging.

The board prevents the execution of certain illegal instructions from crashing the system by monitoring interrupts to the 6809 and its response to them. If the processor does not respond to an interrupt within 128 clock cycles the board resets the 6809 and asserts a special reset vector. The system can then close down the offending task and resume normal operation. This also limits the length of time that interrupts can remain masked by a user, preventing users from keeping the system from task switching and servicing other users.

To further protect the system, the CPU board supports separate user and system "states" with automatic switching to the system state in response to interrupts and system (SWI) calls. Certain functions and memory areas can only be accessed in the system state, preventing unauthorized accesses.

The GMX CPU III also includes a full function time-of-day clock with year and automatic leap year/daylight savings time correction, and a 2K scratchpad RAM; both with battery back-up. To provide precision timing functions, a 6840 PTM with a separate 500 KHz. precision (.0025%) time base oscillator is included. The oscillator is easily user replaceable to provide other time base frequencies (750 KHz. maximum). The single EPROM socket will accept 2K, 4K or 8K EPROMS, with a maximum of 4K mapped into the system address space at any one time. Software switching is implemented by selecting the upper or lower half of an 8K EPROM under hardware or software control.



OS-9 GMX III Operating System

OS-9 GMX III is an enhanced OS-9 Level II that takes full advantage of the features of the GMX CPU III. As a result, the system is faster, more memory efficient, and a more secure multi-user, multi-tasking operating system than the original OS-9 GIMIX II, while retaining complete software compatibility. Throughput is enhanced by the memory to memory DMA and the automatic task switching, while the memory attributes and illegal instruction trapping protect the system and individual users from each other. Sharable system modules in RAM are write protected to prevent tampering. Memory mapping in 2K segments and the ability to load modules in non-contiguous RAM provide more efficient memory utilization. Each task can be allocated a full 64K of RAM, with no operating system overhead in the tasks address space.

UniFLEX for the GMX 6809 CPU III and Intelligent I/O boards is in development.

GIMIX inc 1337 West 37th Place • Chicago, Illinois 60609 • (312) 927-5510 • TWX 910-221-4055

Forum SixtyEight 68

STAFF

Publisher/Editor in Chief
William Sias

Publisher's Assistant
Diane Wright

Graphic Arts Manager
George Sias

Typographer
Barbara Bectel

Circulation Manager
Amanda Morris

Advertising Manager
Arv Sias

Cover Photographer
Philip C. Brautigam

Cover Models
The Silicon Kids

Forum Sixty-Eight (ISSN pending) is published monthly by REMarkable Software, Inc., 1853 Ruddiman Drive, North Muskegon, MI 49445. Telephone (616) 744-4796. Subscription rates for the U.S. are: $21.00 for one year, $39.00 for two years and $57.00 for three years. Application to mail at Second-Class Postage Rates is pending at Muskegon, MI and at additional mailing offices. POSTMASTER: Send address changes to Forum Sixty-Eight Subscriptions, 1853 Ruddiman Drive, North Muskegon, MI 49445.

Forum Sixty-Eight welcomes editorial material of a non-trivial nature and offers generous compensation for the same. All submitted material should be directed to:

Forum Sixty-Eight Editor
1853 Ruddiman Drive
North Muskegon, MI 49445

In This Issue:

Advertiser's Index

About The Cover

This month's cover depicts the "army" of 68000 computers that seem to be appearing from every manufacturer. It looks like the rest of the world is beginning to believe what we've known was true all the time. Motorola Micros are superior! The "ants" may be invading but its still "our" picnic.

OOPS

Its with great embarrassment that I present this issue. As you can see the quality of print is far below our usual standard. I can only hope that the Compugraphic service people are half as ashamed as I am.

Editorial

by Bill Sias

I really hate moving! It seems like every year I have to do it though, I guess that's the price of growth. For your records please change both our Post Office Box and Street Address FROM:

P.O. Box 1192
Muskegon, MI 49443

or

1781 Fifth St.
Muskegon, MI 49441

TO:

1853 Ruddiman Dr.
North Muskegon, MI 49445

Also our phone number has changed from (616) 728-9100 to (616) 744-4796. This change is effective immediately.

The problem with publishing is that you are always working in the future. Right now it's July and we're working on the August Forum Sixty-Eight, which comes out the end of the cover month, and the September Color Computer News, which comes out the end of the month preceeding the cover date.

The reason I bring this up is because I want you to be aware that I fully intend to produce a much larger magazine than the number of pages that are included in this issue. Start-up time is such that we're putting this issue together before you have had a chance to receive the last one. Therefore, we haven't received many articles from you yet. I'm extremely interested in all types of articles. In addition to articles I'm looking for folks to write columns about 68000, Unix, Flex, UniFlex, OS9, C, a column about Heathkit's Hero, and a Languages column. These columns should be tutorial in nature without being "dry". To apply for any of these please send your first article with an outline of the direction you intend follow and explain what you can do that is different (better) than the average magazine column. I'm also interested in columns and/or articles about hardware construction projects. Forum Sixty-Eight pays quite well for all published articles, you won't get rich but you may make enough for that neat new software or hardware item you've been wanting.

I recently had a reason to fire-up my old 8080 CPM based computer. Talk about dinosaurs! Unless you've had an opportunity to work with CPM you really can't appreciate the ease and flexability that the 68XX based disk operating systems have. It never fails to amaze me that the SS-50 buss and its relatives never caught on as well as the S-100 buss did. It's even more amazing when you concider that the 100

this time it "gave up the ghost" just as we hit deadline. Rest assured we'll have "old un-reliable" up and running next issue.

pin connectors were originally selected because they were available as surplus items and therefore cheaper than the 88 pin connectors that the designers wanted, which were available only as new parts. Not only are the disk operating systems better, but the hardware is more reliable (at least in my experience) and generally all the software is superior. Primarily the majority of 68XX software is in the area of languages and utilities, not so much for business usage, with some obvious exceptions. It's about time that someone realized the huge potential for business software in the 68XX marketplace. I'm certain that a major factor in the small number of business only SS-50 computers is the fact that people are becoming aware that the correct way to buy a business computer is to first find the software that will do the job you need done and then buy the computer it was written for. The days of convincing people that the hardware is superior and then expecting them to do anything useful on their own are over. As the 68000 become more popular it will become even more important for there to be quality business software available. I've recently heard excuses that it's too difficult to learn a new processor and the cost of a 68000 computer is prohibitive right now, which is true, but with languages like C, one version of which is reviewed in this issue, neither excuse holds water.

I guess the bottom line is that if we want this part of the industry to keep pace with the rest we have to provide what potential customers need, software.

I recently discovered that Radio Shack has finally put OS-9 and BasicO9 in their store manager's parts books. OS9 is part number 26-3030 and is priced at $69.95 it includes the DOS, editor, assembler and the debugger. Part number 26-3036 is BasicO9 and is priced at $99.95. With Radio Shack making OS9 their "official" 6809 operating system that system is now most certainly the "standard" 6809 operating system if for no other reason that number of users.

Speaking of standards, if standards are set by the largest number of users then the Color Computer ROM Pack port is the "standard" 6809 buss!

Radio Shack's acceptance of OS-9 certainly changes the complection of the Motorola micro market. I feel certain that this will cause much more, and better, 6809 software to be available at lower prices. It is also important for us to not look down upon the new generation of 68XX users as they graduate up to a superior operating system. Who knows, they may even move up to larger computers as they discover the hidden power of the 6809. This acceptance of OS-9 by Radio Shack will have some very interesting impacts on the 6809 world. I don't have any real figures, but I would guess that the current SS-50 market is probably about 30,000 strong. Most of these 30,000 people are using either Flex, Uniflex or OS-9. With OS-9 now on the Color Computer the number of OS-9 user's could easily go to 100,000 in a very short period of time. This means larger profits for software people almost immediately IF they can produce OS-9 software that these newer people will buy. From my experience as publisher of Color Computer News I can tell you that these people are not nearly as uninformed as I've had occasion to hear people say.

I hope you'll forgive the print quality this month. We have an older Compugraphic typesetter attached to one of our computers. It's been flaky ever since its first service call from Compugaphic and this time it "gave up the ghost" just as we hit deadline. Rest assured we'll have "old un-reliable" up and running next issue.

LOOK ! WHAT WE HAVE FOR YOU

FHL Color FLEX	FLEX	$69.95
DBASIC	FLEX	$30.00
EDitor	FLEX	$50.00
	OS-9	$125.00
ASM	FLEX	$50.00
ED/ASM (Purchased as a package)	FLEX	$69.95
A/BASIC COMPILER	FLEX or OS-9	$150.00
DYNASOFT PASCAL	FLEX	$59.95
	W/RUNTIME SOURCE	$89.95
	OS-9	$69.95
	W/RUNTIME SOURCE	$99.95
CRASMB	FLEX or OS-9	$200.00
	CPU Modules	$35.00
	Modules w/source	$70.00
DYNACALC	FLEX	$200.00
DYNASPELL	FLEX or OS-9	$199.00
DYNASTAR/DYNAFORM	FLEX or OS-9	$275.00
STYLOGRAPH	FLEX or OS-9	$295.00
FHL FLEX COLOR UTILITIES	FLEX	$50.00
	W/SOURCE	$75.00
FHL EXTENDED USE UTILITES	FLEX	$49.95
	W/SOURCE	$69.95
AUTOTASK	FLEX	$79.95
	W/SOURCE	$129.95
MCOMMAND.(FREE with purchase of AUTOTASK)	FLEX	$50.00
	W/SOURCE	$75.00
OSM	FLEX or OS-9	$125.00
DYNAMITE +	FLEX	$100.00
SUPER SLEUTH	FLEX or OS-9	$99.00
PL-9 by Graham Trott	FLEX	$198.00
MACE by Graham Trott	FLEX	$98.00
6502 - 6809 TRANSLATOR	FLEX	$75.00
	OS-9	$85.00
6800-6809 AND 6809 PIC/PID TRANSLATORS	FLEX	$50.00
	OS-9	$75.00
6805 AND 6502 SIMULATORS (DEBUGGING TOOLS)	FLEX	$75.00
UNIFLEX SIMULATOR	FLEX	$100.00
OS-9 SIMULATOR	FLEX	$101.00
EDITDISK	OS-9	$79.95
TOOLKIT #1	FLEX	$49.95
	W/SOURCE	$69.95
TOOLKIT #2	FLEX	$49.95
	W/SOURCE	$69.95
HELP	FLEX	$29.95
	W/SOURCE	$49.95
VDISK	FLEX	$100.00
SOME COMMON BASIC PROGRAMS	FLEX	$69.95
SOME PRACTICAL BASIC PROGRAMS	FLEX	$69.95
RMS RECORD MANAGEMENT SYSTEM	FLEX	$200.00
INFOMAG	FLEX	$295.00
BILLPAYER SYSTEM	FLEX	$169.95
READTEST	FLEX	$54.95
	W/SOURCE	$74.95

WE ARE ALSO DEALERS FOR TECHNICAL SYSTEMS CONSULTANTS, MICROWARE, UDRI, AND COMPUTERWARE!!

FOR MORE DETAILED INFORMATION ON THE PROGRAMS WE HAVE AVAILABLE FOR YOUR FLEX OR OS-9 SYSTEMS, CALL OR WRITE FOR OUR CURRENT CATALOG!!

FHL FRANK HOGG LABORATORY

THE REGENCY TOWER • SUITE 215 • 770 JAMES ST. • SYRACUSE, NY 13203
PHONE(315)474-7856 • TELEX 646740

POWERFUL TOOLS FOR ANY PROGRAMMER!

NEW! FROM THE JBM GROUP

ADLIB - a utility which simplifies the maintenance of source code by permitting common data definitions and source code routines to be stored in a single library file. ADLIB is used to pre-process just before compilation. This improves user control by providing a shared definition of file descriptions, record layouts, sections of code and text. The result is that changes, made in one place, can be automatically applied to all your programs.

Available for OS-9 - **$50.00**

LDMAC - consists of a group of subroutine modules callable from BASIC09* to perform routines commonly used in commercial programming or to access OS-9 system services. In some cases, these subroutines duplicate system functions, but do so with greatly reduced system overhead. In other cases, the functions could be done by a BASIC09 routine, but only at the expense of longer code and CPU run time. Some of these routines implement functions that no currently available BASIC09 code or OS-9 utility performs.

Available for OS-9 - **$95.00**

LOOKUP/SLOOKUP - LOOKUP is a directory lookup program providing the ability to retrieve information on a single file or on a group of files matching a wild card discription. It may be used to search a single directory or that directory plus all of its subdirectories. The option of displaying the actual number of disk blocks used versus the number of disk blocks in use is helpful in finding wasted disk space. All numbers are displayed in decimal format rather than hexadecimal to simplify use without being familiar with hex.

SLOOKUP, LOOKUP's companion program, provides the directory information sorted in alphabetical order by file name.

Available for OS-9 - **$75.00**

SORTC - This sort package processes large volumes of data. It is easy to use because of its user-oriented special features. These include:

- A performance predictor function which responds to user controlled parameters.
- A simple-to-use code generator.
- Allows specification of memory allocated to data.
- Creation of user modifiable source code.
- Ascending/descending sequences permitted on multiple keys.
- Support for six standard data types.

The self-contained compounding function sums specified numeric fields and consolidates data records during the sorting process. This simplifies report generation and master file updates as well as reducing disk access and usage.

Available for OS-9 - **$150.00**

CRASMB - MACRO CROSS ASSEMBLER from LLOYD I/O for FLEX and OS9

CRASMB is a macro-conditional cross assembler. It uses machine language overlays or modules called "CPU Personality Modules"to do the work of mnemonic look up. It has directives and other extended commands that are not found in other assemblers. It generates OS9 or FLEX binary files.

FEATURES:

* Cross assembles 8 CPU types:

  Motorola 6800-2-8, 6801-3, 6805, 6809
  Mostek 6502
  RCA 1802
  INTEL 8080-5
  ZILOG Z-80

* 2 passes to generate object code
* library file calls nestable to 12 deep
* conditional assembly nestable to any depth
* macros nestable to any depth, with parameters
* variable length symbols up to 32 characters
* 2048 maximum symbols
* automatically generated labels and symbols
* errors tell file name and line number
* object code format for OS9, FLEX, or neither

For those users who write programs for FLEX and/or OS9, this assembler will allow you to support your source code files on one system. CRASMB can generate OS9 or FLEX formated binary object code files under either disk system.

This program is the most powerful assembler on the market today allowing the programmer to use a single computer system as a development system for so many processors. The user may purchase the source code for the CPU Personality Modules (CPM's) so that it may be modified to create a new assembler for a processor not yet supported.

Written for 6809 FLEX and OS-9

FLEX and OS-9 **$200.00**

CPM's (CPU Modules) **$ 35.00**
w/ source **70.00**

SPECIAL: Purchase CRASMB with all CPU modules w/source for FLEX or OS-9 **$499.00**

A/BASIC Compiler - This BASIC compiler generates pure, fast efficient 6809 machine code for easy to write BASIC source programs. Uses ultra-fast integer math, extended string functions, boolean operators and run-time operations. Output is ROMmable and RUNS WITHOUT ANY RUNTIME PACKAGE. Supports IF-THEN-ELSE structure, random access and several improvements over the 6800 version sold by Microware. Optimized for the 6809, ABASIC is 8 to 10 times faster than the original 6800 version and produces code approximately 30% smaller. (Does not compile RS BASIC or any other BASIC. It is integer only, does not support floating point.

Written for 6809 FLEX or OS-9 **$150.00**

* OS-9 and BASIC09 are registered trademarks of Microware, Inc.
FLEX is a registered trademark of Technical Systems Consultants

FRANK HOGG LABORATORY

THE REGENCY TOWER
770 JAMES ST. - SYRACUSE, NY 13203
TELEX 646740 - (315) 474-7856

NEW! FOR OS-9 and FLEX

DYNA-C

FROM THE AUTHOR OF DYNASTAR II AND DYNASOFT PASCAL !!

Dynasoft sets a new price/performance standard with Dyna-C: a new Small-C derivative unlike all the others. Dyna-C produces compact, ROMable, position-independent, re-entrant, OPTIMIZED code that runs circles around anything in its price class. "C" for yourself:

	Compile/load time(sec)	Compiled bytes	Total bytes	Execute time(sec)	Price $
Introl	219	222	1005	22	375.00
Dyna-C	58	238	970	30	99.95
Wordsworth 2.0	88	307	924	48	99.99
Duggers	89	385	1361	58	120.00
Intersoft	207	490	5573	97	100.00
Telecon	?	?	13484	43	200.00

(All timings under FLEX9 at 1 MHz using sieve benchmark from Sept 1981 Byte. Our own results or as reported in various reviews in '68 Micro Journal)

Dyna-C supports a large subset of standard C, including all statement types, most operators (including ?: and ,), and all data types except float, long, unsigned, struct, multidimensional arrays and bitfields. It goes from your source to executable binary in two quick steps: the one-pass compiler with built-in pre-processor AND OPTIMIZER produces assembly code which is assembled straight to binary using any standard 6809 assembler (including TSC's ASMB, Microware's ASM, Lloyd I/O's OSM and FHL's ASM). While this means maintaining libraries in assembler source form it actually saves disk space and time by eliminating the loader step. It also supports separate compilation so you can split large programs or build your own libraries from C. Source code for the entire runtime system is supplied, so you can customize to your own needs. Requires 36K of user RAM.

OS9: **$109.95**
FLEX9 (including Coco): **$99.95**
RS DOS version coming soon!!

Call or write for our current catalog and receive detailed information on the products we have to offer you for your FLEX or OS-9 systems!!!

THE REGENCY TOWER • SUITE 215 • 770 JAMES ST. • SYRACUSE, NY 13203
PHONE (315)474-7856 • TELEX 646740

Common BASIC09 Subroutines

by John Michaelson

When I first started using OS9 I felt it had three serious short-comings. There were:
1. There is no "mail-box" software for the Level II multi-user system.
2. There isn't a Basic09 command that limits the length of keyboard input.
3. Cursor addressing isn't nearly as easy as it was with my old memory mapped video system.

Since I was fairly certain that none of these problems were going to be solved in the near future by Microware, I decided to attempt to solve them myself. The results of these attempts are the listings at the end of this article. I had several additional problems, the first and biggest was, that since I was going to have to write all of my own software I needed these functions first. Seconds was that, at the time I didn't understand OS9 well enough to patch the language or operating system (I still don't think I know it well enough). Third, I was still learning the language. Far too often you are expected to understand a new operating system simply because you're a programmer. My relief came from the fact that Basic09 can call other Basic09 procedures easily (although at first I didn't think it was so easy).

The first procedure I wrote was Inkey (listing #1). The object of this procedure was to accept a string of a specified length at a pre-determined location on the terminal (if that sounds a bit like Cobol remember that old habits die slow). The calling procedure simply executes a statement that calls the procedure passing the necessary parameters (x and y, maximum length of the string, true or false on clearing the screen and the variable to return the string in). All of the variables are to be determined by the calling program. The theory is that you have already "drawn" a mask on the screen that holds the prompts necessary for the information that the program requires, the operator then "fills in the blanks". For example, a mailing list program could look like:

Mail List

Name _____
Address _____
City _____
State __ Zip Code _____
Telephone (___) ___-____

The operator the "moves" around the screen with whatever keys you feel are appropriate (I use ^U for up, ^d for down, ^R for right, and ^L for left). At first Inkey and Cursor were included in all of the procedures I used. As I

but as the man said," If it ain't broke, don't fix it".

learned more about Basic09 it became obvious that, not only was this practice dumb, but when I changed terminal types it became a pain to edit and recompile all of the programs. I then made them separate procedures. This not only allows easier changes but conserves memory since I can now just keep one copy in memory and let all the users share them.

Variables

The variables used by these programs are fairly self-explanitory. I personally like to use variable names that make it clear exactly what the variable holds and why it was selected. Besides, what good is a structured language if you use unstructured variables? The first two, x_location and y_location, are simply the x and y coordinates of where you wish the "input" to take place on the screen. Max_length is the maximum length you will allow the returned variable to be. Clear_screen is a boolean (true/false) variable that instructs the procedure of when to clear the entire screen.

I think you'll find these procedures quite useful in your programming, especially if you write for operators that are unfamiliar with computers. I've found it to be quite important to maintain control of the operators rather than have the operators control the programs.

I'm certain that if I were to rewrite these procedures I'd do them quite differently. But they've proven to be so useful that I've never bothered.

The memo procedure is another that I'd rewrite if I didn't use it so often. But, it works, and as the man said,"If it ain't broke don't fix it". The program isn't elaborate and I was almost convinced to not submit it. Upon use you'll discover lots of changes that need to be made so please feel free to do so. Some suggestions I'd

With these and other procedures, I've since written, I really feel that I've overcome OS9's shortcomings.

make to anyone modifying it is to have the calling procedure file identify who the user is. Secondly, I'd have it read the date from DATE$. If you do make it better please share it with the rest of us that need an OS9 mailbox system.

Although none of these programs are elaborate, there have served my needs adequately. With these three and several other procedures I've since written I really feel that I've overcome OS9's short commings. Now I'd just like to know how to send an immediate message to an active terminal.

```
PROCEDURE Inkey
 0000      (* Inkey Procedure)
 0013      PARAM x_location,y_location:BYTE
 001E      PARAM Max_Length:BYTE
 0025      PARAM clear_screen:BOOLEAN
 002C      PARAM Entered_Data:STRING[35]
 0038      DIM cursor:STRING[5]
 0044      RUN Mail_Cursor(x_location,y_location,clear_screen,cursor)
 0050      IF Max_Length=0 THEN
 0069        END
 006B      ENDIF
 006D      DIM char:STRING[1]
 0079      Entered_Data=""
 008C      char=""
 0087 10   GET #0,char
 0093      IF char=CHR$(8) THEN
 00A0        char=""
 00A7        Entered_Data=LEFT$(Entered_Data,LEN(Entered_Data)-1)
 00B7        GOTO 10
 00BB      ENDIF
 00BD      IF LEN(Entered_Data)=Max_Length THEN
 00CB        PRINT CHR$(7)
 00D0      ENDIF
 00D2      IF char=CHR$(13) THEN
 00DF        WHILE LEN(Entered_Data)<Max_Length DO
 00ED          Entered_Data=Entered_Data+" "
 00F9        ENDWHILE
 00FD        GOTO 11
 0101      ENDIF
 0103      Entered_Data=Entered_Data+char
 010F      char=""
 0116      GOTO 10
 011A 11   PRINT cursor;
 0123      PRINT Entered_Data;

PROCEDURE cursor
 0000      (* Mail List Cursor Control )
 001D      PARAM x_location,y_location:BYTE
 0028      PARAM Clear_Screen:BOOLEAN
 002F      PARAM cursor:STRING[5]
 003B      DIM Move_Cursor:STRING[5]
 0047      DIM Cls:STRING[2]
```

```
0053      DIM Line:BYTE
005A      DIM Column:BYTE
0061      LET Line=0.
0069      LET Row=0
0072      LET Move_Cursor:=CHR$($7E)+CHR$($11)
00A7      (* )
00AB      (* Change the following for other terminals)
00D7      LET Line=x_location
00E0      LET Row=y_location
00EA      LET Cls:=CHR$($7E)+CHR$($1C)
00F9      LET Move_Cursor:=Move_Cursor+CHR$(Row)+CHR$(Line)
010D      IF Clear_Screen THEN
0116        PRINT Cls;
011C      ENDIF
011E      IF x_location<>0 AND y_location<>0 THEN
0131        PRINT Move_Cursor;
0137      ENDIF
0139      LET cursor=Move_Cursor
```

```
PROCEDURE Memo
 0000     (* Internal Electronic Mail System *)
 0025     PARAM user:STRING[6]
 0031     DIM Recipient:STRING[6]
 0030     DIM Author:STRING[6]
 0049     DIM Sent:STRING[5]
 0055     DIM recieve:STRING[5]
 0061     DIM Message:STRING[322]
 0060     DIM Letter:STRING[344]
 0079     Recipient=" "
 0086     Author=" "
 0093     Sent=" "
 00A0     recieve=" "
 00AD     Message=" "
 00E8     (* *)
 00ED     DIM Pony_Express:BYTE
 00F4     DIM This_Letter:REAL
 00FB     DIM Action:STRING[1]
 0107     (* *)
 010C     CHD "/h0/Office_Mail"
 011F     OPEN #Pony_Express,"Memos":UPDATE
 012F     SEEK #Pony_Express,0
 0138     This_Letter=-1
 0141     (* Main Loop *)
 0150     WHILE NOT(EOF(#Pony_Express)) DO
 0158       This_Letter=This_Letter+1
 0167       GET #Pony_Express,Letter
 0171       Recipient=LEFT$(Letter,6)

 017C       (* Your Mail? *)
 018C       IF Recipient=user OR Recipient="ALL   " THEN
 01A6         Author=MID$(Letter,7,6)
 01B3         Sent=MID$(Letter,13,5)
 01C0         recieve=MID$(Letter,18,5)
 01CD         Message=MID$(Letter,23,LEN(Letter)-23)
 01DF         IF recieve<=MID$(DATE$,4,5) THEN
```

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEX software prices include maintenance for the first year.

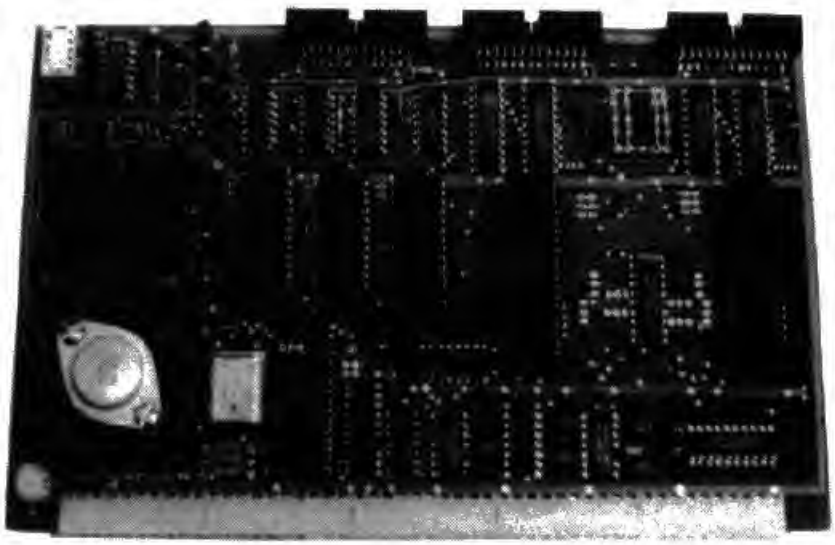
```
01EF            PRINT " "
01F4            PRINT "Message #: "; This_Letter
0207            PRINT "To: "; Recipient
0213            PRINT "From: "; Author
0221            PRINT "Sent:"; Sent
022E            PRINT "Effective: "; recieve
0241            PRINT "Message: "; Message
0252            PRINT " "
0257            PRINT "Any Reply? ";
0267            INPUT Action
026C            IF Action="Y" OR Action="y" THEN
0281              INPUT "To: ",Recipient
028D              WHILE LEN(Recipient)<6 DO
029A                Recipient=Recipient+" "
02A6              ENDWHILE
02AA              INPUT "From: ",Author
02B8              WHILE LEN(Author)<6 DO
02C5                Author=Author+" "
02D1              ENDWHILE
02D5              INPUT "Sent: ",Sent
02E3              INPUT "Effective: ",recieve
02F6              INPUT "Message: ",Message
0307              SEEK #Pony_Express,This_Letter
0311              Letter=Recipient+Author+Sent+recieve+Message
0329              PUT #Pony_Express,Letter
0333            ENDIF
0335            IF Action="N" OR Action="n" THEN
034A              INPUT "Delete this message? ",q$
0367              IF q$="Y" OR q$="y" THEN
037C                Recipient="DELETE"
0389                Letter=Recipient+Author+Sent+recieve+Message
03A1                SEEK #Pony_Express,This_Letter*SIZE(Letter)
03B2                PUT #Pony_Express,Letter
03BC              ENDIF
03BE            ENDIF
03C0          ENDIF
03C2        ENDIF
03C4      ENDWHILE
03C8      INPUT "Care to leave a memo? ",Action
03E6      IF Action="Y" OR Action="y" THEN
03FB        INPUT "To: ",Recipient
0407        WHILE LEN(Recipient)<6 DO
0414          Recipient=Recipient+" "
0420        ENDWHILE
0424        INPUT "From: ",Author
0432        WHILE LEN(Author)<6 DO
043F          Author=Author+" "
044B        ENDWHILE
044F        INPUT "Sent: ",Sent
045D        INPUT "Effective: ",recieve
0470        INPUT "Message: ",Message
0481        Letter=Recipient+Author+Sent+recieve+Message
0499        PUT #Pony_Express,Letter
04A3        CLOSE #Pony_Express
04A9      ENDIF
```

Multi-Port Board

the last I/O board you'll ever need to buy!

$275

- ***3 serial*** (ACIA) I/O ports!
- ***4 parallel*** (two PIA's) I/O ports!
- Room for ***two 6840*** timer chips (6 timers)—Sockets & support logic included.
- ***Addressable anywhere*** from $E000 thru $FF20.
- ***Extended addressing*** supported! (i.e. can be located on any of the 16 pages).
- On-board ***baud rate generator*** with selectable baud rates!
- ***Gold connectors*** insure reliability!
- Manual with schematics included.
- For standard SS-50.



**Computerware's® Multi-Port Board allows you to assemble a complete computer system on a small 50-pin motherboard—just add CPU and memory!*

WAR-

Computerware® declares war against unnecessarily high software prices (especially for OS-9)!

	SSB	FLEX	OS-9
Random BASIC with Sort, Edit, Print Using, On-Error, 6 character variables, 11 digit precision.	$100	$100	$125
Scribe Editor* extensive word & character manipulation—perfect with Macro Assembler!	$ 60	$ 60	$ 75
Macro Conditional Assembler* w/XREF, library files, repeat sequences, any size source file.	$ 75	$ 75	$100

****Far superior to Microware's for a lot less!***

COMPUTERWARE®

Box 668 • Encinitas, CA 92024

(619) 436-3512

Computerware® is a registered trademark of Computerware

TO ORDER

Add shipping of 2% surface or 5% Air. Calif. residents add 6% tax. VISA or Master Card accepted.

Microware's C

by Aaron Lee

After a long wait Microware is now delivering their C compiler. I really don't know what the holdup was, and now that I have it I really don't care, the wait was worth it.

Before examining anything specific I'd like to relate some observations that I've found to be quite interesting. For example examine the following programs.

```
DIM x:INTEGER
x=0
WHILE x<30000 DO
  x=x+1
ENDWHILE
```

```
#include <stdio.h>
main()
{
  int x;
  x=0;
  while (x<30000) {
    x++;
  }
}
```

The first is a Basic09 program and the second a C program, both of which merely count to 30,000. The first executes in 14.29 seconds and the second in .87 seconds. A rather significant difference by anyone's definition. However, if you change the two programs to these:

```
DIM x:INTEGER
x=0
WHILE x<30000 DO
  x=x+1
  PRINT x; " ";
ENDWHILE
```

```
#include <stdio.h>
main()
{
  int x;
  x=0;
  while (x<30000) {
    x++;
    printf("%d ",x);
  }
}
```

the situation changes quite drastically. The second now runs in 3:29.15 minutes and the first in 2:59.18 minutes. Quite an interesting change of events. I'm quite certain that its possible to change the results by some bit of code that I simply don't know. Perhaps C's I/O is really slower than Basic09's.<g>

As with any language, there are several ways to do any task. I tryed the following variations for testing purposes:

```
#include <stdio.h>
main()
```

THE COMPLETE 6809

The Microware C
is a fully implemented
Kernighan and Ritchie
standard C Compiler
with relocatable
assembler, linker, profiler,
UNIX* and OS-9*
standard libraries.

Suggested U.S.
retail price $400.

Now available from

MICROWARE
5835 Grand Avenue
Des Moines, Iowa 50312
515-279-8844
Telex: 910-520-2535

*UNIX is a trademark of Bell Laboratories. OS-9 is a trademark of Microware and Motorola, Inc.

After a couple of sessions I discovered that C is more like a "shorthand" BASIC09 than a new language.

```
{
  int x;
  for (x=0; x<30000; x++)
  ;
}

#include <stdio.h>
main()
{
  int x;
  x=0;
  while (x<30000) {
    x++;
  }
}

#include <stdio.h>
main()
{
  int x;
  x=0;
  do {
    x++;
    } while (x<30000);
}
```

a second interesting point I noticed was that the three programs executed at so nearly the same speed that I couldn't detect any speed difference. However, the last version generated less object code as evidenced by this abbreviated directory:

```
bytecount name
--- ----------
1C7 count_while
1C1 count_do
1C3 count_for
```

Obviously the differences are extremely small but so are the programs.

One of the most important parts of any language is the documentation. Microware has really out done themselves on their C manual. It contains not only the usual "This is how to copy the disk" but examples on how to call C procedures from Basic09 programs. Since there are some differences on how each of the two languages handle some variables this is really important information (see the listings at the end of this article). The explanations are quite complete and the examples are accually useful. One of my pet peeves are example programs that are only useful to prove a point, "Type them in and throw them away", the programs in the manual are easily modified to be usable in real life situations. I've never seen a Microware manual of this quality before, I certainly hope this is an indication of future Microware manuals.

As good as the manual is I don't think that it alone is enough to teach yourself C. I'd recommend that you pick up "C Programming Guide" by Jack Purdum published by Que Corporation, 7960 Castleway Drive, Indianapolis IN 46250. Microware includes Kernigham and Ritchie's "The C Programming Language" which for many people may be sufficient to learn C.

My first impression when I began learning C was that it was difficult and quite foreign but after a couple of sessions I discovered that C is more like a "short hand" Basic09 than a new language. You may discover that by keeping this attitude the learning process may go faster.

Some unusual (I'm not sure that's really the word I want) features of Microware's C are:

1. A Real-time profiler. By adding -p to the compiler command line you will add code to your program to count the number of times each procedure is called. This feature will allow you determine which procedures are being called most often, thereby allowing you the opportunity to spend your time optimizing the procedures that are using the most CPU time.
2. The complete package includes a relocatable linking assembler. I was assured that the linking loader and relocatable assembler will be available

On the plus side, Microware's C contains a new storage class called the direct class.

separately soon. The advantage of this arrangement is that all of the modules produced will be position independant and reentrant (both are requirements for OS9 memory modules).

There are, however, some differences between Microware's C and the C specification in Kernighan and Ritchie's "model" C. These are:

1. Bit fields are not supported.
2. Constant expressions for initializers may include arithmetic operators if the operands are of type char or int.
3. Assignment operators must be of the "newer" type (i.e. +=, *=) not the older forms as: =+, =*.
4. #if <constant expression> is not supported. However, #ifdef, #ifndef, #else and #endif are.
5. Macro definitions and strings are limited to one source line in length.
6. New-line (/n) refers to a line feed in The C Programming Language, in Microware's C /n is a carriage return.

As you can see these exclusions are all items that can be easily "programmed around" or really don't make much difference.

On the plus side, however, Microware's C contants a new storage class that should more than make up for the loss of anything above. This new class is called the "direct" class. This storage class takes advantage of the 6809's use of the direct page of memory. Essentially this new class allows you to specify up to 255 bytes of variables that will be stored on the direct page of memory. This allows much faster variable access time which means faster running programs.

```
PROCEDURE sorter
DIM i,n,d(100):INTEGER
n=1000
i=RND(-(PI))
FOR i=1 TO n
  d(i):=INT(RND(1000))
NEXT i
```


continued on page 35

```
PRINT "Before:"
RUN prin(1,n,d)
RUN qsort(d,n)
PRINT "After:"
RUN print(1,n,d)
END

PROCEDURE prin
PARAM n,m,d(100):INTEGER
DIM i:INTEGER
FOR i=n TO m
  PRINT d(i); " ";
NEXT i
PRINT
END

#define swap(a,b) { int t; t=a; b=t; }

/* qsort to be called by BASIC09:
     dim d(100):INTEGER any size INTEGER array
     run cqsort(d,100) calling qsort.
*/

qsort(argcnt,iarray,iasize,icount,icsiz)
int argcnt;        /* BASIC09 argument count */
    iarray[],      /* Pointer to BASIC09 integer array */
    iasize,        /* and it's size */
    *icount,       /* Pointer to BASIC09 (sort count) */
    icsiz;         /* Size of integer */
{
    sort(iarray,0,*icount);  /* initial qsort partition */
}
/* standard quicksort algorithm from Horowitz-Sahni */
static sort(a,m,n)
register int *a,m,n;
{
     register i,j,x;

     if(m < n) {
          i = m;
          j = n + 1;
          x = a[m];
          for(;;) {
               do i += 1; while(a[i] < x); /* left partition */
               do j -= 1; while(a[j] > x); /* right partition */
               if(i < j)
                    swap(a[i],a[j])        /* swap */
               else break;
          }
          swap(a[m],a[j]);
          sort(a,m,j-1);                   /* sort left */
          sort(a,j+1,n);                   /* sort right */
      }
}
```

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a *new-generation* "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell *anywhere* on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted. "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited *only* by available memory.
ONLY $250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total *Management/Marketing Planning* system. Its built-in select/sort/merge module — and optimized machine code — makes it superfast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.
ONLY $500

TMP FreeForm fills the void between a structured DBMS and word-processing; it's an "electronic index card" package with an endless list of valuable applications such as parts cataloging, look-up inventories and listings of *any* kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.
ONLY $150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word-processors, BASIC and other high-level languages.
NO CHARGE with TMP/Manager

TMP T.M.

TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals *further* assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are *incomparable* in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD *TODAY*

Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back-up; performance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

SMOKE SIGNAL
Chieftain Computers

31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

NCC 83 Report

by Bill Sias

The National Computer Convention was held in Anaheim at the Disneyland Hotel and the Anaheim Convention Center. The exhibits were contained in three buildings. It was quite unfortuate that the majority of the SS-50 activity was in a tent-like structure that closely resembled an oven, at least as far as temperature is concerned. But even with 118 degree heat neither Smoke Signal Broadcasting nor GIMIX had any problems with heat failure. The same wasn't true for other manufacturers.

I'm certain that the heat caused a great lack of sales for the SS-50, as well as others, people. However, there wasn't a lack of excitement over VAR/68 system from Smoke Signal and the GMX III from GIMIX.

The VAR/68 is an SS-50 "desk-top" computer from Smoke Signal. Its main features are a cabinet that is much smaller that the usual SS-50 cabinet, and it incorporates just the SS-50 section of the motherboard (I believe it has 8 slots) rather than a section of 50 pins and an I/O buss with 30 pins.

The GMX III is a radically new implementation of the 6809 that requires (at this time) OS9 level III. Some of the features of the GMX III are intelligent I/O processors and a new CPU board. The GMX III was reviewed last issue from a business standpoint.

I would be less than honest if I said I enjoyed conventions. I do like meeting the people I've known only by their telephone voice and the time I can spend "talking computers" with people that are well informed. But I do dislike the attitudes of people who attack anyone with a press pass. The vast majority of the exhibits at conventions don't do a thing for me. There either "old tech" or they simply don't relate to either my business or personal interests.

The best times I had at NCC were when I was trying to find out what was on the drawing board at various companies. Its amazing, to me, the number of 68000 computers that will be available by January. There's everything from SS-50 to S-100 to single board 68000s coming out.

I find it quite interesting that, while "everyone" is claiming that CPM is the wave of the future, all of the new computers use either UNIX or a UNIX look alike for their operating system. I guess programmers have finally had enough and are demanding better tools to work with.

Although I've spent far too much space complaining about conventions, I'll still see you in Des Moines. That's my sort of convention, all 6809 68000 and a chance to "talk computers" with the big kids.

Reader I/O

Dear Forum,
I certainly hope you'll allow some controversy in Forum Sixty-Eight. A magazine isn't much good unless it allows you to get mad once in a while. Good old fashioned temper tantrums are what magazines should be for.

Sincerely,
Lee Nagerski

* If anger is what you want I'm certain I've got just what you need:
I spelled your name wrong on purpose!

Best wishes for the new magazine. I certainly like the business and engineering thrust you have chosen for it.

Robert Harris

* I do too, but its only you as readers that can make it go that way. Forum Sixty-Eight is just what the name implies, a forum for users of microprocessors that carry 68 as the first two digits. You will be the people to guide it. My opinions or directions are no more valid than yours.

Dear Forum,
I'm currently a subscriber to your other magazine, Color Computer News. Through those pages I've really outgrown BASIC and perhaps my Color Computer. I don't have the finances to buy an SS-50 system so I guess I'll keep it. I've heard that the Color Computer running either FLEX or OS9 is as powerful as a GIMIX or Smoke Signal anyway. What limitation do I actually face with my 64k Color Computer.

John Parsons

* The limitations are really processor speed and expandability. Your Color Computer is effectively stuck at less that 1 MHz and 64K of memory, while all the SS-50 computers that I'm aware of are rather easily expandable to 1 or more Megabytes of memory and the processor speed is 2 MHz. However, the newest GMX III systems are being delivered with the CPU running at 2.25 MHz. My office GIMIX has the 2.25 MHz CPU board and has been running great. There are some other things that make the difference such as reliability, Both GIMIX and Smoke Signal use gold plated connectors whereas the Color Computer's connectors are tin plated. Please don't misunderstand, I think the Color Computer for the price, but I've never pretended to run my business on one. I'm certain it would be possible to but the inconvience would be simply too much.

Assembly Line

by Frank Hoffman
19535 NE Glisan Street
Portland, OR 97230

We just moved to our new office in down town Portland. Mount Hood is visible through the window which opens to the east. I'm sitting here just thinking of how to start out again. It's been almost two months since the first. NCC has come and gone, the OS9 seminar is coming up, and I've just had a birthday.

I'm going to review NCC for a little while. It was the first one I've attended. I think NCC is the largest of its kind within this star system. There is so much to see at a show like this it's almost impossible to see the whole thing without a plan of "attack". The only thing I objected to was the rather warm ("hot") temperatures. Oh, well, it goes with the territory.

Almost every manufacturer and software house either attends or exhibits at these shows. I found it very interesting to see what the rest of the computer world is doing or is about to do.

I managed to meet many of the fine people from the SS-50 world. This was very beneficial to me because meeting a person face to face makes it easier to trust them. Most of us from the SS-50 bunch gathered every night at the GIMIX hospitality suite for munchies and chat. This proved to be the best way to get detailed information on some of the new software and hardware currently on the market. One thing you don't want to do is have Richard Don from GIMIX introduce you to someone you've never met or seen before. He has a lot of fun confusing you.

These are some of my impressions from NCC '83. It really was a lot of fun, though you would be surprised what the other guy looks like. Well, lets get back to the subject.

This time I'll cover more assembler directives and macros. We'll also get started on the library routines mentioned last time.

MORE A.L. DIRECTIVES

Last time I covered the ORG, END, RMB, EQU, and FCC directives. To start off we'll describe the assembler's page formating and control directives. The following is a list of these directives.

1. OPT option list
2. SPC count expression
3. PAG
4. TTL string
5. NAM string
6. STTL string
7. LEN count expression

"This is a title" is covered up by the use of the NAM directive since both TTL and NAM are two names for the same directive.

8. INI expression list

The OPTion directive is used to switch one or more flags either on or off. The two simplest flags are the LIS (for listing on) and PAG (for page formating on). The reverse forms are NOL and NOP.

Examples:

```
OPT PAG
OPT NOL
OPT NOP
OPT LIS
```

When the PAG option is turned on the assembler handles the listing of the program by pages. It prints a title consisting of one or more lines, the date, and the page number. Then roughly 50 to 55 lines of the formated source lines with the object code are printed. Finally, the page is ejected by CR,LF pairs or a form feed to the next page.

If the listing is turned off by the NOL switch, then lines are listed only when errors are detected. Some assemblers use different forms of these option switches and some are very useful when debugging a program.

The SPC (for space) directive generates a gap of pure carriage return,line feed pairs. This function is used to make a program a little easier to read because it makes a wide border area between two programs, subroutines, or similar code. However, it eats up a lot of paper. A better border is a line of dashes or asterisks.

Example:

```
* ++++++++++++++++++
*
*  a program here
*
* ++++++++++++++++++
```

The PAG directive (not to be confused with the option OPT PAG) is a means to cause a form feed to occur. This is a nice way to start a program listing. This directive would be used after the titles and option flags are set.

Example:

```
TTL First title
STTL Sub title
OPT PAG
PAG
```

The PAG directive does not work when the page formating option is turned off by the OPT NOP directive.

The TTL and NAM directives set the current page title. Most assemblers allow up to 40 characters. Enough room on the line must exist for the page number and assembler name. The LLOYD I/O and TSC assemblers use the NAM and TTL directives for the same title area. The STTL (sub-title) directive sets second or sub-title string. I use the NAM directive for the main program name and the STTL directive for the current file name or subroutine name.

Examples:

```
TTL This is a title
NAM DO VERSION 1.1
STTL BY JOHN DOE
```

The results look like this:

```
DO VERSION 1.1 LLOYD I/O ASSEMBLER PAGE 1
BY JOHN DOE    March      15
```

In the above example the TTL string "This is a title" is covered up by the use of the NAM directive since both TTL and NAM are two names for the same directive.

Most of the newer assemblers keep track of the next available addresses for both the object code and the storage for variables.

The LEN directive or a similar directive is used to set the current line length which can be printed on your printer. Some assemblers even allow setting the page length, the form feed character or string, and how close to come to the bottom of the page. This allows you to set up a listing for different printers and sizes of paper.

Examples:

```
LEN 80
LEN 132
LEN 40
```

The INI or similar directive allows you to send out characters which have a value equal to an expression. I use this function to set my printer in the condensed print format so I can use 132 characters per line on the 8.5 inches wide paper. There is one catch though with a program that uses an INI directive. When it is assembled with the output sent to another device such as the terminal, the INI characters may do some unexpected function such as clearing the screen.

Examples:

```
INI 27,31
INI 27,30
```

When you set your printer to use the condensed print don't forget to change the line length. Your listing will look a little strange with half the page used.

The next group of directives generate bytes of object code. We covered some of them last month. These directives either control the address to where object code is stored or actually generate code.

1. ORG
2. RAM
3. RMB
4. END
5. FCB
6. FDB
7. FLW
8. FCC
9. FCS

The most important directive in an assembly program is the ORG directive. This is because it tells the assembler where your program is supposed to be stored. I'm including it here again because it belongs in this group of directives.

Examples:

```
ORG $0000
ORG $C100
ORG PGBEG
```

Most assemblers will default to an origin (ORG) of zero ($0000).

The RAM directive is almost the same as the ORG directive. It tells the assembler where your read/write memory starts for variables and your subroutine address stack. Most assemblers don't have this directive. But for the 6809 CPU its a very nice function to have. It is intended to work with the RMB directive and the storage address counter.

Most of the newer assemblers keep track of the next available addresses for both the object code and the storage for variables. To access the current value of the object code address (called the PC or program counter) use the the "*" (asterisk) character in any expression. To access the storage counter in an expression use the "." (period). It is important for the new assembly language programmer to realize that most 6809

Even the high level languages such as "C" and "Pascal" require you to define all variables before the program or the procedure is defined.

programs should be written using the PIC technique. This process uses the "*" and "." counters. I design my programs to define all storage for variables first. This way I know how much memory is being consumed for storage. It forces the program to be PIC because the origin of the storage counter is set to zero ($0000). All storage is accessed by the program as an offset from the U (user stack) register. The U register is made to point to an area of memory which can be used for reading and writing.

There are two reasons for doing this. First the program may eventually be burned into an EPROM. You can't write into an EPROM. The second reason is that OS9 systems are designed to work this way. A program may be loaded at an address one time and at a different address the next time. The same goes for the memory assigned to it for variable storage. Even the high level languages such as "C" and "PASCAL" require you to define all variables before the program or procedure is defined.

The directive RMB is used to allocate or reserve memory bytes for storage. It increments the storage counter "." by the value of the expression associated with the directive. Most programs will define labels using the RMB directive as its variables.

Example:

```
REGA  RMB   1
REGB  RMB   1
REGCC RMB   1
REGDP RMB   1
REGX  RMB   2
REGY  RMB   2
REGU  RMB   2
REGS  RMB   2
REGPC RMB   2
MEME  RMB   2
XXX   RMB   2
ACCL  RMB   8
ACCER RMB   1
ADDDI RMB   1
```

Each use of the RMB directive increases the value of the storage counter by the value of the expression. This allows the storage counter to be ready for the next use of RMB.

The END directive is mentioned here again as it also belongs in this group. Used by itself it has no real meaning except for documentation of the program. However, if it has its optional expression, then the END directive tells the assembler where the program is to start. Under FLEX this is the way to generate the "transfer address". Once a program is loaded into memory for execution, FLEX jumps to the transfer address.

Examples:

```
END START
END WARMS
END BEGIN
END $C100
```

The remaining directives of this group generate object code. They are used to generate constants in the form of character strings, messages, address tables, and look-up tables.

The first is the directive, form constant byte (FCB). It generates a single byte of code which has the value of the expression.

Examples:

```
FCB 0
FCB 4
FCB $30
```

Most assemblers allow multiple expressions for a single FCB to

These two directives are more complex ... or maybe more flexible than the previous directives.

generate multiple bytes of code.

Examples:

```
FCB 0,0
FCB $03,$14,$15,$92,$65,$40
FCB 'T,3
```

The FCB directive is probably the easiest way to generate object code. It is very useful for making special look-up tables. The lower 8 bits of the expressions are used as the value. The upper half is just chopped off.

The form double byte (FDB) directive is used to generate two bytes of object code for every expression associated with it. It uses the whole 16 bit value of each expression.

Examples:

```
FDB 0
FDB START
FDB NAMES-BEGIN
FDB 1400,$1B,$000A,$000A
FDB 60*60*24
```

Some assemblers which use 32 bit integer math instead of 16 bit integer math utilize the directive FLW or something similar to generate four bytes of code from a single expression. It means to form constant long word. (A word is 16 bits and a long word is 32 bits.) The 68,000 CPU assemblers have this directive.

Examples:

```
FLW START
FLW CMDTABLES
FLW BRANCHTABS-START
```

The string generation directives are FCC and FCS. They function almost identically except that FCS sets bit seven (MSB) of the last byte generated. This can be used to signal the end of the string. These two directives are more complex ... or maybe more flexible than the previous directives. They accept multiple expressions and strings. I think some examples will help show what I mean.

Examples:

```
FCC /this is a string with a byte following/,4
FCS /this is two strings/,$D,$A,/with a CR LF pair/
FCC 0,0,$D,$A,$D,$A
FCS /OSM/
```

These examples show most of the complexity involved. Now that we've covered this group I'll show you some examples of look-up tables.

Examples:

a command look up table

```
CMD     FCS     /GOTO/
        FDB     QGOTO-START
        FCS     /GOSUB/
        FDB     QGOSUB-START
        FCS     /ON ERROR GOTO/
        FDB     QONGO-START
        FCB     0               END OF TABLE

OPERS   FCB     '*,4
        FDB     MMUL-START
        FCB     '/,4
        FDB     MDIV-START
        FCB     '+,3
        FDB     MADD-START
        FCB     '-,3
        FDB     MSUB-START
        FCB     0               END OF TABLE
```

The "-START" is used to help make the program PIC. (It makes the value an offset address from the real assembly time address.) At run time the real effective address is calculated from the offset. This type of table addressing works similar to the

Macros are a little hard to understand but if you are carefull to read and then try out what you find here, you'll be doing yourself a real favor.

relative addressing modes. I'll cover this more later.

<i>MACROS<i>

This next section is on the care and feeding of macros. Macros are a little hard to understand but if you are careful to read and then try out what you find here, you'll be doing yourself a real favor.

Macros are a way of storing a sequence of lines for later use. The reason for doing this is that the sequence of lines may have to be used several times. It is kind of easy to make syntax errors when typing in the same series of lines several times in addition to being very time consuming. What you do is make the assembler do the hard part. If you store the sequence of lines away and give it a name then by the use of that name at a later time the assembler starts to recall each line one at a time until the last line of the macro is reached. If an error was made in the definition then the error appears each time the macro name is used.

There are always two parts in using macros. First you define it. Then you can call it. You can not call it then define it. That makes no sense to the assembler. Once you have the macro text defined you can do many things with it. There are several directives that do rather fancy things. But first lets try defining a macro.

```
label   mnemonic    operand    comment
field   field       field      field

PRINT   MACRO
        ENDM
```

This macro contains a single null line. Its name is "PRINT". The directive "MACRO" starts the macro definition. The name is given just as if it were a label, except that it is always limited to six characters. This is because words in the operator field are limited to a maximum of six characters. The assembler always checks labels to see if the word "MACRO" follows. If it does the label is not defined as a label but as a macro name. Once the definition is started all lines up to the directive "ENDM" (end macro) are stored in memory as text lines. These lines are not processed for assembly until the macro is called.

If you were to reset your computer and use the monitor to examine memory you would find your macro definitions stored in memory following their names. Lets try another macro.

```
 label  mnemonic   operand    comment
 field  field      field      field

 PRINT  MACRO
        LEAX       :1,PCR     POINT TO
THE STRING
        LBSR       PSTRNG     PRINT
THE STRING
        BRA        :2         BRANCH
AROUND THE STRING
 :      FCC        "&1",4     DEFINE
THE STRING
 :      EQU        *          BRANCH
LABEL
        ENDM                  END OF
MACRO
```

This macro introduces two new concepts. The first is "parameter substitution" and the second is automatic labels (or a form of local labels). Parameter substitution is a means of telling the assembler that when the macro is called you want to modify it a little bit. This example is of a macro which prints a string. Here are some calls to it with different strings to be printed.

Examples:

```
PRINT "(0)...QUIT"
PRINT "(1)...FORMAT DISKS"
PRINT "(2)...VERIFY DISKS"
PRINT "(3)...COPY   DISKS"
```

When a macro is called as in our four examples they are known to "expand". Another term is "expansion time". It is called this because each line of the macro is read from memory and placed into the assembly line buffer for processing. All parameter substitutions are performed on that line before it is assembled. If a substitution is done it makes the line longer. In our example there is only one line that has a parameter substituted.

Any time the character "&" (ampersand) is detected with a digit of 1 to 9 following, it is deleted. Then a check is made to find the parameter by its number of 1 to 9. If it was not given when the macro was called then nothing else is done. But as in our example parameter number one (&1) is defined and the "&1" is replaced by the string of characters between the double quotes. So the single line containing the parameter substitution characters is "expanded" to:

Examples:

```
FCC "(0)...QUIT"
FCC "(1)...FORMAT DISKS"
FCC "(2)...VERIFY DISKS"
FCC "(3)...COPY   DISKS"
```

The macro can substitute the parameter any number of times. Parameters are separated from each other by a comma. If spaces or commas must be included as a part of a parameter then the parameter is "delimited" by either single or double quotes.

Examples:

```
PRINT 'THIS IS A TEST'
PRINT "AN EMBEDDED COMMA, IS IN THIS PARAMETER"
PRINT NO-COMMAS-OR-EMBEDDED-SPACES
PRINT 'HAVE NO FEAR, CHARLIE IS HERE
```

Now look at what the assembler sees when the macro is expanded with our first example.

```
label  mnemonic    operand    comment
field  field       field      field

       PRINT       "(0)...QUIT"
       LEAX        L00001,PCR POINT TO
THE STRING
       LBSR        PSTRNG     PRINT THE
STRING
       BRA         L00002     BRANCH
AROUND THE STRING
L00001   FCC                  "(0)...QUIT",4
DEFINE THE STRING
L00002   EQU        *          BRANCH
LABEL
       ENDM                   END   OF
MACRO
```

This macro example is set up to generate position independent code (PIC). It also uses automatic labels which are a form of local labels. (This form is available on all LLOYD I/O assemblers. Most relocating assemblers also have it.) The assembler substitutes the letter "L" followed by the ASCII numeric digits representing the value of the automatic label counter. Every time the character ":" (colon) is detected as the first character of a line the counter is incremented by one and the substitution is performed. This is extremely useful in cases like our example.

Expressions may use automatic labels by using the colon as an operand. If the colon is followed by a positive or

negative decimal number, its value is added to the current automatic label counter before the substitution is performed. The number is called an offset. If it is not present then a zero offset is used. By close examination of our example we see that it used both a one (1) and a two (2) as an offset in this manner.

By the way this isn't the only way to make a macro like this work the way it does. If the subroutine "PSTRNG" returns with the index register "X" pointing to the byte following the $04 then the following macro definition will work. Notice that it does not use any automatic labels.

```
label   mnemonic    operand     comment
field   field       field       field

PRINT   MACRO
        LEAX        *+8,PCR     POINT TO
THE STRING
        LBSR        PSTRNG      PRINT THE
STRING
        JMP         0,X         BRANCH
AROUND THE STRING
        FCC         "&1",4      DEFINE THE
STRING
        ENDM                    END OF
MACRO
```

Most assemblers which support macros can use this last example with the right "PSTRNG" subroutine. You can help yourself a lot by knowing your system software. I use a set of macros for program debugging which print ASCII messages and byte and word values in hex and decimal. They look like this:

```
label   mnemonic    operand     comment
field   field       field       field

PRINTN  MACRO
        LEAX        *+8,PCR  POINT TO THE
STRING
        LBSR        PSTRNG  PRINT THE STRING
        JMP         0,X         BRANCH AROUND THE
STRING  FCC         "&1",4      DEFINE THE STRING
        LEAX        &2           POINT TO NUMBER
        LBSR        OUTDEC      PRINT AS DECIMAL
        ENDM                    END OF MACRO
```

The "OUTDEC" subroutine can be any routine which prints the two bytes pointed to by the "X" register as a sixteen bit decimal number. Of course these two new lines could print anything useful. Parameter number two must be an indexed addressing mode because the instruction "LEAX" only allows indexed addressing modes. Here are some calling examples:

Examples:

```
PRINTN 'LOOP COUNTER      = ',"CNTER,U"
PRINTN 'CONDITION COUNT   = ',"LEVEL,U"
PRINTN 'FALSE LEVEL       = ',"SWIT,U"
PRINTN 'LABEL VALUE       = ',"0,X"
```

Notice that parameter two has an imbedded comma. The quotes are required in order to allow the commas to be a part of the parameter. The indexed addressing mode must use the comma.

Here is an idea that may work for some of you who are interested in this type of work. Design a pseudo central processing unit and instruction set using macros. Add some sophisticated instructions like multiply and divide with 16 or 32 bit integers. Design it to use a good set of indexed addressing modes. Then begin to design all of your programs using this new assembly language. Later on when you want to change computers with a different CPU and bring across your programs you simply redefine the macros to reflect the new CPU instruction set. Then you reassemble your programs. This is one way to make a "compiler"!

For now I'm going to start the

It's much easier to change a single subroutine than changing many lines or even many files.

library routines that I promised last month. We'll start with the disk file accessing routines. This is the outline:

1. open file for read
2. open file for write
3. read a line/byte from an open file
4. write a line/byte to an open file
5. close a file

All other operations can be done with these routines except renaming a file. The examples used here assume that the programs using these routines are written in position independent code (PIC). Therefore all variables are defined first and are accessed as an offset from the user stack pointer ("U"). The following examples are for FLEX(tm).

INSERT FLEX LISTING

The next subroutines are for OS9. They are the same disk input/output routines. Again the examples used here assume that the programs using these routines are written in position independent code (PIC).

INSERT OS9 LISTING

The OS9 subroutines are much shorter. Most of the time you don't even need the short subroutines, except for one point. It's much easier to change a single subroutine than changing many lines or even many files. The examples show good programming practices in that they are well documented with comments and input/output expectations. Some programs may not want to use the "ERROR" subroutine as a means of recovering from errors. You would have to write your own "error trap" to handle those special cases.

Here's how to use the subroutines. First open a file for read. Assuming that the file name is in the FLEX line buffer and that the line buffer pointer is pointing to the first character of the name.

Example:

```
LEAX    FCB1
LBSR    GETNAM NAME THE FILE
LBSR    ROPEN  OPEN THE FILE FOR READ
```

If the file is to be opened for writing it must not exist.

Example:

```
LEAX    FCB1
LBSR    GETNAM  NAME THE FILE
LBSR    WOPEN   OPEN THE FILE FOR
                  WRITING
```

That's for FLEX, now for OS9.

Example:

```
LBSR    ROPEN OPEN THE FILE FOR READ
STA     RPATH
```

Again if the file is to be opened for writing it must not exist.

Example:

```
LBSR    WOPEN OPEN THE FILE FOR WRITING
STA     WPATH
```

Now lets say that we want to read lines of text from disk until the end of the file is reached.

Example:

```
LINE    RMB     256     TEXT LINE BUFFER
FCB1    RMB     320     FILE CONTROL
                          BLOCK
*
        LEAX    FCB1
        LBSR    GETNAM  GET NAME OF FILE
```

SOFTWARE DEVELOPMENT TOOLS from **LLOYD I/O**

ED

A Programmer's Editor for FLEX and OS9

ED is a combination "Screen" and "Line" editor with extended features that allows the programmer to write programs for any computer language or even word processing files such as letters and manuscripts. ED operates identically on FLEX and OS9.

FEATURES:

* edits files larger than memory
* extended disk file functions
* restart editing with out reloading
* automatic start up procedure file
* redefineable keyboard
* works with any terminal
* calculator for integer expressions
* command macros with parameters

OSM

6809 MACRO ASSEMBLER for FLEX and OS9

OSM is a MACRO Assembler with CONDITIONAL Assembly directives and other extended commands that are not found in other assemblers. If you write programs for OS9 and/or FLEX then OSM is for you because it generates OS9 or FLEX machine language binary files under OS9 or FLEX.

FEATURES:

* Motorola 6809 standard mnemonics
* library file calls nestable
* conditional assembly nestable
* macros nestable, with parameters
* up to 32 character symbols
* automatically generated labels
* errors tell file name and line
* object code format for OS9 or FLEX

ORDER FROM: **LLOYD I/O**
19535 NE GLISAN STREET
PORTLAND, OR 97230

TELEPHONE: **[503] 666-1097**

VISA AND MASTER CHARGE ACCEPTED

CRASMB

MACRO CROSS ASSEMBLER for FLEX and OS9

CRASMB is a MULTI-CPU Cross Assembler. This is our best assembler. CRASMB is a macro-conditional assembler. It uses machine language overlays or modules called "CPU Personality Modules" to do the work of mnemonic look up. It has directives and other extended commands that are not found in other assemblers. It generates OS9 or FLEX binary files.

FEATURES:

* Cross assembles 8 CPU types:
  Motorola 6800-2-8, 6801-3, 6805, 6809
  Mostek 6502, RCA 1802, INTEL 8080-5
  ZILOG Z-80
* Manufactur's standard mnemonics
* library file calls nestable
* conditional assembly nestable
* macros nestable, with parameters
* up to 32 character symbols
* automatically generated labels
* errors tell file name and line
* object code format for OS9 and FLEX

Other Products From LLOYD I/O:

ASM (FLEX 6809 Macro Assembler)
ISM (FLEX 6809 Interactive Assembler

*** PRICING ***

name	FLEX	OS9
ED (editor)	**$50**	**$50**
CRASMB (cross assembler)	**$200**	**$200**
each CPU module binary	**$35**	**$35**
with source code	**$70**	**$70**
All eight w/ source	**$499**	**$499**
OSM (OS9/FLEX assembler)	**$99**	**$99**
ASM (FLEX assembler)	**$50**	n/a

ISM [interactive ass.] free with FHL Flex

LLOYD I/O
19535 NE GLISAN STREET
PORTLAND, OR 97230

Next month I'll really spend some time with the rest of the basic programmer library package.

```
        LBSR    ROPEN   OPEN THE FILE
LOOP    EQU     *       LOOP POINT
        LEAX    FCB1
        LEAY    LINE
        LBSR    RLINE   READ A LINE
        BNE     LOOPER  ERROR SO EXIT
                          LOOP
*
        BRA     LOOP
*
LOOPER  EQU     *
        CMPB    8       IS IT AN END OF
                          FILE
        LBNE    ERROR   NO
        LBSR    CLOSE   YES
```

Well this is a start on our library of subroutines for disk file input and output. Once you have a decent set of subroutines in your library it will always be easier to write new programs.

NEXT TIME

Next month I'll really spend some time with the rest of the basic programmers library package. I'll start showing how to use macros and conditional assembly directives to do many of the tasks involved in calling these subroutines. This will be a step closer to a "higher" level assembly language. Until then.

continued from page 19

approach that "This computer-friendly liveware is protected by Byte Bat".

The Byte Bat is the creation of Jae Evans. As head of the company that wrote the flight software of the Harier jet, and other major custom software assignments, Jae has experienced a recurring desire to hit the machines with which he works.

All too often, computers aren't up when you need them, or some sort of system error cost you a lot of time and effort. Hitting a computer or terminal with anything substantial can be a satisfying but expensive act, so I came up with the Byte Bat", Evans says.

The Byte Bat is being distributed through computer dealers nationwide, at a suggested retail price of $9.95. Frustrated computer users who are frustrated even more when they find that their local dealer isn't carrying the Byte Bat can order their very own for $12.50 postpaid by calling (800) 227-3900 ((800) 632-2122 in California).

Byte Bat is a trademark of MicroTie Systems Corporation.

```
* ------------
* SYSTEM ERROR EXIT
*
*       INPUT:                  (X)=ADDRESS OF FILE CONTROL BLOCK
*       OUTPUT:                 NONE
*       ERROR OUTPUT:           NONE
*
ERROR   EQU     *               REPORT SYSTEM ERRORS
        JSR     RPTERR          PRINT ERROR MESSAGE
        JSR     FMSCLS          CLOSE ALL FILES
        JMP     WARMS           GOTO FLEX
*
* ------------
* GET FILE NAME
*
*       INPUT:                  (X)=ADDRESS OF FILE CONTROL BLOCK
*                               FLEX BUFFER POINTER AT FIRST
*                               CHARACTER OF FILE SPECIFICATION
*                               (BUFREG EQU $CC14)
*       OUTPUT:                 (X)=ADDRESS OF FILE CONTROL BLOCK
*                               FILE NAME AND EXTENSION IN FCB
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
*
GETNAM  EQU     *
        JSR     GETFIL          GO GET FLEX FILE SPEC.
        BCS     ERROR           IF ERRORS, REPORT
        LDA     1               DEFAULT EXTENSION TO "TXT"
        JSR     SETEXT
        SEZ                     RETURN EQUAL STATUS
        RTS
*
* ------------
* OPEN FILE FOR READ
*
*       INPUT:                  (X)=ADDRESS OF FCB
*                               FILE ALREADY NAMED
*       OUTPUT:                 (X)=ADDRESS OF FCB
*                               FILE OPEN FOR READING
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
*
ROPEN   EQU     *               READ OPEN
        LDA     1               SET STATUS
        STA     0,X             FOR READ OPEN
        JSR     FMS             DO OPEN
        LBNE    ERROR           EXIT ON ERROR
        RTS                     RETURN
*
* ------------
* OPEN FILE FOR WRITE
*
*       INPUT:                  (X)=ADDRESS OF FCB
*                               FILE ALREADY NAMED
*       OUTPUT:                 (X)=ADDRESS OF FCB
*                               FILE OPEN FOR WRITING
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
```

```
*
WOPEN   EQU     *               WRITE OPEN
        LDA     2               SET STATUS
        STA     0,X             FOR WRITE OPEN
        JSR     FMS             DO OPEN
        LBNE    ERROR           EXIT ON ERROR
        RTS                     RETURN
*
* ------------
* READ A TEXT LINE
*
*       INPUT:                  (X)=ADDRESS OF FCB
*                               (Y)=ADDRESS OF LINE BUFFER
*       OUTPUT:                 (X)=ADDRESS OF FCB
*                               (Y)=ADDRESS OF LAST CHARACTER
*                                   WHICH WOULD BE A (CR) $0D
*       ERROR OUTPUT:           (Z) BIT CLEAR
*                               (B) ERROR NUMBER
*
RLINE   EQU     *
        BSR     RCHAR           READ A CHARACTER
        BNE     RLINEE          EXIT IF EOF
        CMPA    $0D             IS IT (CR)
        BEQ     RLINEX
        CMPA    $20             LIMIT STORING TO SPACE THROUGH
        BLO     RLINE           A DEL CHARACTER ($20 TO $7F)
        CMPA    $7F
        BHI     RLINE           LIMIT
        STA     0,Y+            STORE THE CHARACTER
        BRA     RLINE           AND INCREMENT THE POINTER
RLINEX  EQU     *
        STA     0,Y             SAVE THE EOL CHARACTER
        SEZ                     RETURN STATUS OK
RLINEE  EQU     *               EXIT
        RTS
*
* ------------
* READ A CHARACTER
*
*       INPUT:                  (X)=ADDRESS OF FCB
*       OUTPUT:                 (X)=ADDRESS OF FCB
*                               (A)=CHARACTER READ
*       ERROR OUTPUT:           (Z) BIT CLEAR
*                               (B)=ERROR NUMBER
*
RCHAR   EQU     *
        JSR     FMS             READ A CHARACTER
        BNE     RCHARE          IF NO ERROR THEN RETURN
        RTS                     RETURN NO ERROR STATUS
RCHARE  LDB     1,X             GET ERROR NUMBER
        CLZ                     CLEAR Z BIT
        RTS                     RETURN WITH ERROR STATUS
*
* ------------
* WRITE A TEXT LINE
```

```
*
*       INPUT:                  (X)=ADDRESS OF FCB
*                               (Y)=ADDRESS OF LINE BUFFER
*       OUTPUT:                 (X)=ADDRESS OF FCB
*                               (Y)=ADDRESS OF LAST CHARACTER
*                                   WHICH WOULD BE A (CR) $0D
*       ERROR OUTPUT:           (Z) BIT CLEAR
*                               (B) ERROR NUMBER
*
WLINE   EQU     *
        LDA     0,Y+            GET A CHARACTER
        CMPA    $0D            IS IT (CR)
        BEQ     WLINE0          WRITE EOL ANYWAY
        CMPA    $20            LIMIT STORING TO SPACE THROUGH
        BLO     WLINE           A DEL CHARACTER ($20 TO $7F)
        CMPA    $7F
        BHI     WLINE           LIMIT
WLINE0  PSHS    A               SAVE THE CHARACTER
        BSR     WCHAR           WRITE A CHARACTER
        PULS    A               RECALL CHARACTER AGAIN
        BNE     WLINEE          EXIT IF ERROR
        CMPA    $0D            IS IT (CR)
        BEQ     WLINEX          EXIT IF (CR)
        BRA     WLINE           AND INCREAMENT THE POINTER
WLINEX  EQU     *
        SEZ                     RETURN STATUS OK
WLINEE  RTS                     EXIT
*
* ------------
* WRITE A CHARACTER
*
*       INPUT:                  (X)=ADDRESS OF FCB
*       OUTPUT:                 (X)=ADDRESS OF FCB
*                               (A)=CHARACTER READ
*       ERROR OUTPUT:           (Z) BIT CLEAR
*                               (B)=ERROR NUMBER
*
WCHAR   EQU     *
        JSR     FMS             WRITE A CHARACTER
        BNE     WCHARE          IF NO ERROR THEN RETURN
        RTS                     RETURN NO ERROR STATUS
WCHARE  LDB     1,X             GET ERROR NUMBER
        CLZ                     CLEAR Z BIT
        RTS                     RETURN WITH ERROR STATUS
*
* ------------
* CLOSE A FILE
*
*       INPUT:                  (X)=ADDRESS OF FCB
*       OUTPUT:                 (X)=ADDRESS OF FCB
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
*
CLOSE   EQU     *               CLOSE A FILE
        LDA     4              GET CLOSE STATUS
        STA     0,X             SET IT
        JSR     FMS             DO CLOSE
```

```
        LBNE    ERROR
        RTS                     EXIT WITH NO ERROR STATUS
*
*
*
```

```
* ------------
* SYSTEM ERROR EXIT
*
*       INPUT:                  (B)=ERROR NUMBER
*                               (C) BIT SET
*       OUTPUT:                 NONE
*       ERROR OUTPUT:           NONE
*
ERROR   EQU     *               REPORT SYSTEM ERRORS
        OS9     F$EXIT          EXIT PROGRAM
*
* ------------
* OPEN FILE FOR READ
*
*       INPUT:                  (X)=ADDRESS OF FILE NAME
*       OUTPUT:                 (X)=POINTS PAST FILE NAME
*                               (A)=PATH NUMBER
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
*
ROPEN   EQU     *               READ OPEN
        LDA     1               SET STATUS FOR READ MODE
        OS9     I$OPEN          DO OPEN
        LBCS    ERROR           EXIT ON ERROR
        RTS                     RETURN
*
* ------------
* OPEN FILE FOR WRITE
* CREATE A FILE
*
*       INPUT:                  (X)=ADDRESS OF FILE NAME
*       OUTPUT:                 (X)=POINTS PAST FILE NAME
*                               (A)=PATH NUMBER
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
*
WOPEN   EQU     *               WRITE OPEN
        LDA     2               SET ACCESS MODE TO WRITE
        LDB     3               SET FILE ATTRIBUTES FOR R/W
        OS9     I$CREA          DO CREATE AND OPEN
        LBCS    ERROR           EXIT ON ERROR
        RTS                     RETURN
*
* ------------
* READ A TEXT LINE
*
*       INPUT:                  (X)=ADDRESS OF TEXT LINE
*                               (A)=PATH NUMBER
*       OUTPUT:                 (Y)=NUMBER BYTES READ
```

```
*       ERROR OUTPUT:           (C) BIT SET
*                               (B) ERROR NUMBER
*
RLINE   EQU     *
        LDY     255           LIMIT TO 255 CHARACTERS
        OS9     I$RDLN         READ A LINE
        RTS                    EXIT
*
* ------------
* WRITE A TEXT LINE
*
*       INPUT:                  (X)=ADDRESS OF TEXT LINE
*                               (A)=PATH NUMBER
*       OUTPUT:                 (Y)=ACTUAL NUMBER BYTES WRITTEN
*       ERROR OUTPUT:           (C) BIT SET
*                               (B) ERROR NUMBER
*
WLINE   EQU     *
        LDY     255           LIMIT TO 255 CHARACTERS
        OS9     I$RDLN         READ A LINE
        RTS                    EXIT
*
* ------------
* CLOSE A FILE
*
*       INPUT:                  (A)=PATH NUMBER
*       OUTPUT:                 NONE
*       ERROR OUTPUT:           EXITS VIA ''ERROR''
*
CLOSE   EQU     *               CLOSE A FILE
        OS9     I$CLOS          DO CLOSE
        LBCS    ERROR           EXIT ON ERROR
        RTS                     EXIT WITH NO ERROR STATUS
*
```

Smoke Signal's New VAR/68 Series Miros Designed for System Integrators.

A new computer series, designed specifically for Value Added Resellers (VARs), is being announced by Smoke Signal as their entry into the desktop computer market.

The new computers, called the VAR/68 series, come standard with 128Kb of RAM, 8 serial ports, 1 parallel port and an ergonomic video display and keyboard. The VAR/68 is now available in configurations with: dual 750 Kb floppies; floppy and 5Mb or 20Mb Winchester; and floppy with two 20Mb Winchesters. Tape streamer backup is also available.

VAR/68 computers are easily upgraded. All systems can handle 1Mb of RAM, 16 (or more) serial ports, 4 parallel ports, special function boards -- such as communications and graphics, and hard disk storage up to 150 Mb (and higher).

Available with the VAR/68 is OS9-Level II, a UNIX-like multi-user, multi-tasking operating system. Features like password protection, record level lock-out, and dynamic memory allocation afford the VAR maximum flexibility and performance in multi-user applications.

Resellers can choose among a structured BASIC with Pascal type data structures, as well as COBOL, Pascal and C compilers.

To allow a VAR to concentrate on his specialty vertical application, Smoke Signal also offers standard accounting software packages, word processing, the TMP (Total Management Planning) family of integrated spread sheet and data base management software as well as a library of application software such as medical office and CPA packages. The VAR/68 is available with a 2-4 week delivery time at end-user prices starting at $4,325 with dual floppies, up to $10,585 with a floppy and dual 20Mb Winchesters.

Documentation has also been taken to a higher level with both Audio and Video Cassette learning tapes. With this "New Tech" form of operator training the transition of putting the new system "on-line" becomes much faster and easier.

For further information, contact Don Simonsen at Smoke Signal, (213) 889-9340, or write Smoke Signal at 31336 Via Colinas, Westlake Village, CA 91362.

GIMIX, Inc.
1337 W. 37th Place
Chicago, IL 60609

Announcing a NEW Super High Performance Winchester Disk

3 to 4 Times Faster Access Than Current Hard Discs plus 2-1/2 Times More Storage Capacity

GIMIX proudly announces their new 47 Megabyte 5.25" Winchester drives. Designed to complement both their GMX III system as well as their previous computers, the new 47 Megabyte drive will sell for only $2200 more than GIMIX's previously available 19 Megabyte Hard Drive.

GIMIX recently announced that their intelligent I/O processors (IOPs), currently available only for their GMX III computer, will soon be available for Level II systems. The IOP's software includes routines for controlling an "intelligent" modem for both auto answer and auto dial. Possibilities for such a system are endless and include things such as automated remote order entry. Prices to be announced.

GIMIX Inc
1337 West 37th Place
Chicago, IL 60609

Radio Shack Introduces
10 New Multi-User Programs

For Office Applications
On TRS-XENIX

Ten new computer programs specially designed for multi-user TRS-XENIX systems are now available at Radio Shack computer centers and participating Radio Shack stores and dealers. Radio Shack is a division of Tandy Corporation.

GENERAL LEDGER (26-6201), offered for $599, is a "total" system that will interact with the Radio Shack Accounts Payable, Accounts Receivable and Payroll program.

GENERAL LEDGER is suitable for all types of businesses, including service firms, merchandising or manufacturing firms. The user may define and print a variety of financial statements, including a balance sheet (with or without last year's comparative figures), an income statement (for one or all profit centers or departments, and with or without budget or comparative figures), cash flow, statement of changes in financial position and analysis of working capital.

PAYROLL (26-6203) is a complete system that calculates and prints checks and distributes to General Ledger accounts. It is offered for $699.

ACCOUNTS RECEIVABLE (26-6204) is an open item and/or balance forward system that will interface with the General ledger, Order Entry/Inventory Control and Sales Analysis programs. Accounts Receivable automatically calculates sales tax, late charges, due date of invoice, discount date and amount. It is offered for $599.

ACCOUNTS PAYABLE (26-6205), offered for $599, is an accrual system that will interface with General Ledger. It automatically calculates discount date and amount, prints aged open item reports and allows full or partial payment of invoices.

Both single-pass invoicing and two-pass open order entry with separate billing is easly accomplished with Order Entry/Inventory Control (26-6207), available for $599. It calculates sales tax and commissions and prints invoices, purchase advice reports, back order reports, picking tickets and more.

ORDER ENTRY/INVENTORY CONTROL requires and interacts with the Radio Shack Accounts Receivable program. It requires three disk drives or hard disk.

SALES ANALYSIS (26-6208), available for $399, allows examination of sales information with data obtained from customers' accounts receivable files or the item files of Order Entry/Inventory Control. It requires three disk drives or hard disk.

MULTIPLAN (26-6480), offered for $349, is a second generation spreadsheet analysis program that can display multiple data windows on the screen simultaneously.

High level implementation of ANSI-74 COBOL standard can be accomplished with COBOL Development System (26-6455), available for $699.

BASIC INTERPRETER (26-6457), offered for $299, allows developing applications in the BASIC language.

Development of multi-user software is possible with TRS-XENIX Development System (26-6401), available for $750. The software package includes a "C" language compiler, communications, text processing and electronic mail.

Tandy Corporation Introduces Videotex and Office Information System (VIS)

A New Microcomputer Information Storage, Retrieval and Distribution System

Videotex and Office Information System (VIS) is a new information storage, retrieval and distribution system from Tandy Corporation. A hardware and software package, VIS is designed to allow quick and easy access to large data bases stored in the desktop TRS-80 Model 16B Computer.

VIS is a two-way interactive system that transmits information electronically. Any terminal or computer with terminal capability may be used to receive and display text and, if compatible, graphics and computer programs.

VIS offers state of the art technology at prices that are a fraction of the cost of comparable systems. Some features, such as multiple keyword/synonym access, are not available on any other system.

Data is established and organized by the information providers who define their user population, sell or provide information and information updates, and establish any security access rules. A system operator maintains the computer hardware and VIS software for either one or more information providers.

As a private internal videotex system, VIS can meet the requirements of an information storage and delivery system within a company's internal operations, including daily schedules, policy files or as a bulletin board.

Using a computer in an office or a portable terminal from thousands of miles away, for example, VIS could

allow checking of factory orders against usable inventory, corporate sales figures and production capacity, client or patient reports.

As a mass market system, VIS can be used as a "public" data base in which paid subscribers are allowed access to data such as news, weather and stock reports. Airline schedules, real estate multiple listing directories and credit bureau information are just a few of the many services possible with VIS.

A user may request a unit of information, called a "document", by entering the title, or by using one or more "keywords", words used as a cross-reference. VIS is so user-friendly it will accept misspelled words or retrieve alternatives based on phonetic similarities.

Since a user requesting information on two topics could possibly receive more information than needed, VIS features logical operators (and, or, exclusive or) to permit the user to target in on the exact documents required.

Information can also be accessed from "menus", a "user-friendly" operation for those who have little or no experience with data base systems.

All that is required to set up a basic VIS system is a two-disk TRS-80 Model 16B with 384K RAM, a 12-megabyte hard disk system, two modems and VIS software. This system can handle up to 200 incoming calls a day, assuming each call averages between five and 10 minutes each over a 24-hour period.

From this basic configuration, a system can expand as necessary. Ultimately, the system operator can provide service to as many as 256 information providers per system. Additional hard disks can be added until the system could store all the information contained in a complete set of encyclopedias. ASCII text is compressed before being stored on disk so that Radio Shack's 12-megabyte hard disk systems can effectively store 15 megabytes of text.

The VIS software was developed in highly modular form in the "C" programming language, and features an advanced back-up and recovery system. It is designed to be a continuously evolving product, with software enhancements made available to all system operators.

VIS operates with the TRS-80 Communications Multiplexor, for internal or external videotex systems. An eight-port version accepts eight incoming phone lines, while a 16-port version can service up to 16 users simultaneously.

VIS is compatible with virtually all TRS-80 desktop computers equipped with a 300-baud modem. Users may also utilize the TRS-80 Model 100 portable computer with built-in modem or a communications terminal.

The full VIS application software package is offered for a one-time $3,500 license fee. An additional $1,000 is charged for a multiplexor software support module, a device that acts as an "electronic funnel", assembling several information requests coming into the VIS host computer at one time.

For more information contact:

Radio Shack Videotex Department
1700 One Tandy Center
Fort Worth, TX 76102
(817) 390-2642

Radio Shack TRS-80
Model 16B Computer

Features 256K Memory and TRS-XENIX Multi-User Operating System

The new top of the line TRS-80 Model 16B Computer from Tandy Corporation / Radio Shack offers superior performance in a compact desktop system. Now including additional memory and easy user expansion, the Model 16B also comes with the TRS-XENIX multi-user operating system.

Single and dual floppy disk drive versions of the Model 16B desktop computer are offered. Each built-in slim-line 8-inch floppy disk drive given 1.25 megabytes of storage.

The single drive version of the Model 16B (26-6004) is $4,999, and the two disk version (26-6005) is $5,798. Both versions have 256K bytes of internal memory, expandable to 768K in 128K steps. They are available through the more than 400 Radio Shack Computer Centers and participating Radio Shack stores and dealers nationwide.

The Model 16B's dual-processor design features Z-80A and state of the art MC68000 16 / 32-bit microprocessors to permit the high operating speeds and large memory capacity required by many business users.

For greater expandability, the Model 16B features a built in, user accessible card cage. Four plug in expansion slots easily allow addition of a hard disc, additional memory or high resolution graphics.

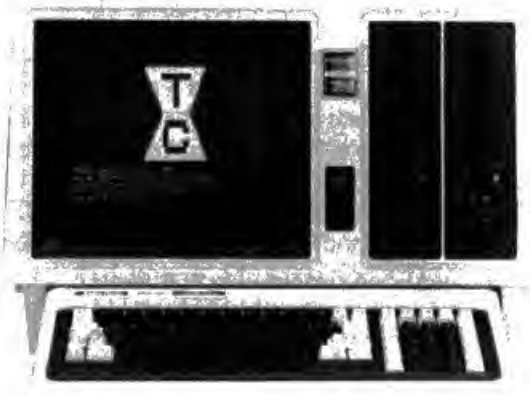
Other features of the Model 16B are a high resolution 12 inch green video display, a detachable keyboard with a numeric datapad, two RS-232C serial interfaces and a parallel printer interface.

A Model 16B, two low-cost Radio Shack DT-1 data terminals (26-6050, $699 each) and a hard disk drive are the only equipment needed for multi-user operation of the Model 16B.

The TRS-XENIX multi-user operating system increases office productivity at a substantial reduction of hardware costs. It allows up to three people to use the Model 16B simultaneously with no loss of performance. Some simultaneous operations may require a minimum of 384K memory.

Using the TRS-XENIX operating system, all data in the Model 16B can be stored on the hard disk. Files can be shared by all users in the system, or restricted to certain users only. Peripherals attached to the Model 16B, such as printers, plotters and modems, can be shared by all system users for maximum savings.

TRS_XENIX is supported by a variety of multi-user software. Model 16B can also run all TRS-80 Model II and Model 12 programs (single-user only).



Conway Publications, Inc.
Peachtree Air Terminal
1954 Airport Road
Atlanta, GA 30341

New On SITENET
A Free On-Line Global Data Base

The new global study, now accessible free of charge through Conway's on-line SITENET data base, identifies some 150 major projects which typically involve investments of $500 million or more, or which have global significance in terms of economic development.

Of the 150 "super projects" compiled and added to his SITENET data base, Conway has visited more than 70 of the sites.

The well-known development engineer and electronic publisher names these 10 projects as the most significant:

1) The Siberia Ob-Pechora Water Project Proposal;
2) The North America Grand Canal Plan;
3) The Atlantic iceberg towing venture;
4) The Netherlands Delta Plan;
5) The SASOL Synthetic fuels complex in South Africa;
6) JUBAIL, the port and industrial complex in Saudia Arabia;
7) The Seikan Tunnel in Japan;
8) The Beijing tunnels;
9) JARI, the former Ludwig Amazon Project;
10) Tokamak, the Princeton fusion facility.

Conway explained the significance of the new SUPERPRO data file and his "top ten" list in an article written for Industrial Development magazine's elite readership of corporate real state executives and economic development officials.

"Clearly, there are times when men gather themselves to build great and extraordinary things. Today, the world may be on the threshold of another era of great developments," explained Conway, who has been gathering data on development projects for 30 years.

Several factors seem to be combining to favor the development of "super projects', Conway noted. Among them are urgent need; advanced technology and recently acquired environmental wisdom.

The first three projects on Conway's "top ten" list deal with redistribution of the fresh water supply from Arctic or Antarctic regions to arid lands.

"Unless there is a dramatic development in reducing the cost of desalting sea-water, some such redeployment of resources will be essential in feeding the world's population in the next century," Conway said.

The Netherlands Delta Plan, item four of Conway's "top ten", is both a striking model for land reclamation and protection and significant lesson in energy recovery.

SASOL in South Africa may well be a prototype for coal utilization in many oil-hungry nations. Saudi Arabia's JUBAIL will provide data on the feasibility of creating new industrial centers.

Japan's Seikan tunnel will be studied in relation to many gaps in the world's transport system, such as the England-France Channel and the Gibraltar Crossing.

The Beijing tunnels, originally developed as a means of affording the world's urban populations some protection from attack, may provide new ideas for conservation.

JARI, the former Ludwig Amazon Project, already has produced such new concepts as the floating industrial plant. Princeton's Tokamak fusion facility holds the hope of a new and abundant source of energy.

"SUPERPRO" AVAILABLE ON-LINE, FREE TO USERS WORLDWIDE

Computer users in over 152 U.S. cities may scan the SUPERPRO file, or any of SITENET's other files, free of charge.

By placing a telephone call to any node on the global network, users may scroll through extensive files of industrial location data, economic development incentives and contacts, and site selection criteria.

SITENET's file of "super projects" is the first in a series of ongoing additions to the growing data base, which began operating around the clock on June 1 of this year. Conway Data, Inc., the publishing and consulting company which operates SITENET, currently is transferring data contaned in its books, periodicals and microfiche onto electronic retrieval system.

Like the new file of "super projects", the rest of SITENET's data fles pertain to economic development worldwide. Included are a broad range of topics related to quality of life factors, disaster risks, demographic data and government incentive programs for industrial growth and economic expansion.

The unique free data base offers a mix of files which are of interest both to the general public and to Conway Data's primary audience of corporate real estate executives, facility planners and area development professionals.

Consisting of data developed in more than a quarter-century of research, consulting and publishing, SITENET offers instantaneus electronic access to Conway's global data base.

To view a directory of local SITENET telephone numbers or a complete menu of network data files, the user simply sets his telephone modem at 300 baud, full duplex, eight bit, no parity. By placing a long distance call to (404) 252-0999, the user hears a high-pitched tone and is then ready to connect his modem and log onto SITENET.

After connecting the modem, the screen will display an upper-case "L" and a question mark. The user enters a carriage return, a period and a second carriage return, with no spaces. A network welcome message will appear, at which time the user enters the letters SIT, followed by a carriage return.

By following on-screen instructions to request the SITENET Index file (IND), the user may find the local network telephone number and place all future calls free of long distance charge. Users who experience problems in logging onto SITENET may call the network's technical support staff at (404) 458-6026.

For Further Information Contact:

Ken Kimsey, Coordinator of Marketing Services, Conway Data, Inc., 1954 Airport Road NE, Atlanta, Georgia 30341-4996, USA. Telephone: (404) 458-6026. Telex: 80-4468ATL.

To have your annoucement appear in this column send copy to:
NEWS FORUM
1853 Ruddiman Drive
North Muskegon, MI 49445

Universal Data Research, Inc.
2457 Wehrle Drive
Buffalo, New York 14221
(716) 631-3011

Universal Data Research, Inc. today announced Professional Video Tape production on a series of training tapes for T.S.C.'s Uniflex system and Universal Data's Database Management & Accounting Software in all popular formats.

For further information contact:
JoAnne Heckman
(716) 631-3011

Advanced Digital Technology
13400 Northrup Way, #27
Bellevue, Washington 98005
(206) 643-2382

INTRODUCING THE EMULATOR THAT IS AS FAST AS YOUR TARGET SYSTEM

Are you tired of your Emulator putting restraints on your target system, or not having the capabilities which would make it really useful? Then discover Advanced Digital's NEW 4009B Test Station. The combined Emulator/Logic Analyzer that is totally transparent. Totally real time. The 4009B meets or exceeds manufacturer's timing specifications. It can even handle VMA cycles for DMA operation and interrupt intensive systems.

The 4009B enables you to emulate 6809 and 6809/E series microprocessors with a stroke of the keyboard. That's right; you make NO software or hardware changes. Up to 128K of memory overlay, four hardware triggers, and 2K deep of trace, help you pinpoint problems faster. A line by line assembler and a powerful trace disassembler are standard. They permit you to view software and make changes without returning to the development system.

MORE THAN AN EMULATOR

As a "digital engineering workstation," Advanced Digital's 4009B gives you a synergistic approach to integration problems. Two microprocessors control the system. One is dedicated to emulation while the second microprocessor supervises the 32 channel Logic Analyzer and all system menus. This design lets you emulate and perform all system functions simultaneously.

ADVANCED DIGITAL HAS PROVEN ITSELF TO MICROWARE, CREATIVE MICRO SYSTEMS, AND OTHER 6809 USERS

Ken Kaplan, President, Microware, "The Emulator has allowed us to solve problems in days that without it could have taken weeks."

Lyle Johnson, Senior Design Engineer, Modular Mining Systems, "In our several years of experience in the microprocessor industry, we have never dealt with a more customer oriented company. Your support of this product is, quite simply, amazing."

Pete Prossen, Owner, Creative Micro Systems, "Our engineers have been fighting over the unit. So you'd better send us another one."

DISCOVER THE MOST POWERFUL DIAGNOSTIC TOOL AVAILABLE FOR HARDWARE AND SOFTWARE INTEGRATION!

Call Advanced Digital Technology for a free demonstration.

Jack DeFriel
National Sales Manager
Advanced Digital Technology

TO REALIZE THE FULL POWER & PERFORMANCE OF THE 6809, LOOK TO GIMIX.

GIMIX OFFERS YOU A VARIETY OF SS50 BUS COMPONENTS AND SYSTEMS.

OS-9 GMX III

The GMX 6809 CPU III and OS-9 GMX III. A Multi-user, Multi-tasking package for the ultimate in System Performance plus protection of the system and other users from crashes caused by errors in individual users programs.
#01 (CPU & Software) ..... NEW! **$1698.01**

INTELLIGENT I/O PROCESSOR BOARDS increase system throughput by reducing interrupts to the host, buffering data transfers, and data preprocessing. Prices include on board firmware. Requires system drivers.
#11 3 port RS232 Serial (SS30) ..... NEW! **$498.11**
#12 4 port Parallell (SS50) **$536.12**
OS-9 GMXIII drivers . . (included when purchased with GMX III package) . . **$200.00**
OS-9 Level 2 users -- contact GIMIX for system requirements and availability.

192K GMX III #79 SYSTEMS: All include GMX 6809 CPU III and OS-9 GMX III (#01); a #11 3 port intelligent serial I/O & cables; #19 Classy Chassis; 192KB Static RAM, #68 DMA controller, all necessary cables, power regulators, and filler plates. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are included.
#79 with dual 40 track DSDD drives **$5098.79**
#79 with dual 80 track DSDD drives ..... NEW! **$6298.79**
#79 with **#88** 8" Dual Drive Disk System **$7698.79**
#79 with **#90** 19MB Winchester subsystem & one 80 track DSDD drive . . **$8998.79**
UniFLEX for the GMX 6809 CPU III and Intelligent I/O boards is in development.

OS-9 GMX I; OS-9 GMX II; FLEX; and UniFLEX

The **#05** GIMIX 6809 PLUS CPU board **$578.05**
Options: GMX DAT ..... **$35.00** SWTPC DAT **$15.00**
9511A ..... **$312.00** 9512 ..... **$265.00**

#49 64KB GHOST SYSTEM includes: #05 CPU; #19 Classy Chassis; 64KB static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX I. You can software select either FLEX or OS-9, The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are also included.
#49 with dual 40 track DSDD drives **$4398.49**
#49 with dual 80 track DSDD drives **$4698.49**
#49 with **#88** 8" Dual Drive Disk System **$5998.49**
#49 with **#90** 19MB Winchester subsystems & one 80 track DSDD drive **$7398.49**

#39 128KB SYSTEM includes: #05 CPUwDAT: #19 Classy Chassis; 128KB of static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX II. You can software select either FLEX or OS-9, The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB, and GMX-VDISK for FLEX are included.
#39 with dual 40 track DSDD drives **$4998.39**
#39 with dual 80 track DSDD drives **$5298.39**
#39 with **#88** 8" Dual Drive Disk System **$6598.39**
#39 with **#90** 19MB Winchester subsystem & one 80 track DSDD drive **$7998.39**
UniFLEX, available at extra cost, requires 8" or Winchester drives. A signed license agreement with TSC is required before shipment.

You can add to any GIMIX system RAM, I/Os and other options, or substitute non-volatile RAM. GIMIX will customize to your needs.

COMING SOON: Contact GIMIX for price and availability on 40MB and 72MB Winchester (5¼") drives, removeable pack Winchesters. 256KB static RAM boards.

All GIMIX systems are guaranteed for 2MHz operation. GIMIX systems include documentation for all boards and software in a GIMIX binder. **ALL DRIVES ARE 100% TESTED AND ALIGNED BY GIMIX.**

ALL BOARDS AND SYSTEMS ARE ASSEMBLED, BURNED-IN, AND TESTED. GOLD-PLATED BUS CONNECTORS ARE USED.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033. Visa or Master Charge also accepted.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX Inc. reserves the right to change pricing, terms, and product specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Choose from GIMIX' wide variety of system components.

The **GIMIX CLASSY CHASSIS #19** consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Mother board with baud rate generator board **$1398.19**
Triple Disk regulator card and cables **$88.22** Baud rate generator card **$88.93**
Missing cycle detector ..... **$38.23** Filler plates **$14.92**
Back panel connector plates (specify) . **$8.60** 50 Hz. option **$30.00**

MEMORIES (GIMIX uses only Static RAM)
#67 64KB NMOS STATIC RAM board **$478.67**
#64 64KB CMOS STATIC RAM board w/battery back-up **$568.64**
#34 8K PROM board **$98.34**
#32 16 socket PROM/ROM/RAM board **$238.32**

I/O Boards (see above for Intelligent I/Os)
#41 Single port serial, RS232/20ma. current loop **$88.41**
#43 2 port serial, RS232 ..... **$128.43**
#46 8 port serial, RS232 ..... **$318.46**
#42 2 port parallel **$88.42**
#45 8 port parallel **$198.45**
#50 serial, RS232, RS422, RS423 ..... **$244.50**
#52 SSDA serial, RS232, RS422, RS423 ..... **$254.52**
#54 ADLC serial, RS232, RS422, RS423 ..... **$268.54**
Each cable with connectors for back panel mounting (specify board) **$24.95**

DISK CONTROLLERS
#68 DMA (featured in all systems above) **$588.68**
#28 dbl. dens. programmed I/O (5" drives only) **$298.28**
#58 single dens. programmed I/O (5" and/or 8" drives) **$226.58**
#48 same as **#58** but for 5" drives only **$198.48**
Cable sets: 8" with Back Panel connector **$29.25**
for two 8" external drives **$44.26**
for two 5" drives **$34.96**

SOFTWARE: GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the **#68** controller.
When ordered with any controller, FLEX is **$30.00**

GMXBUG PROMs and manual **$98.65**
Boot or Video boot PROM ..... **$30.00** UNIFLEX boot PROM **$50.00**
OS-9 GMX I ..... **$200.00** OS-9 GMX II **$500.00**
Editor ..... **$125.00** Assembler **$125.00**
BASIC-09 ..... **$200.00** RUNB **$100.00**

DISK DRIVES FOR GIMIX SYSTEMS -- complete with cables and power regulators.
5" DSDD 40 track ..... 2 for **$900.00**
5" DSDD 80 track ..... 2 for **$1300.00**
#88 8" Dual 8" DSDD drives, cabinet, power supply, & cables **$2698.88**
Cabinet only ..... **$848.18** 220V 50Hz. Option, add **$30.00**
Filler plate ..... **$14.83** Cable for 2 drives **$44.82**
Cable for 4 drives ..... **$67.84** Cable for cabinet to mainframe **$45.81**

WINCHESTER SUBSYSTEMS: for use only in GIMIX systems with **#68** DMA controller.
#90: includes one 19MB drive, interface, and Software **$3588.90**
#91: includes two 19MB drives, interface and Software **$5288.91**
Contact GIMIX for price and availability of other forthcoming subsystems.

OTHER BOARDS
#76 GHOST 80X24 VIDEO BOARD **$396.76**
#66 50 pin Protoboards ..... **$56.66** **#33** 30 pin Protoboards **$38.33**
#03 6800 CPU **$224.03**
#06 6800 CPU with timers ..... **$288.06** Baud rate option, add **$30.00**

#08 RELAY DRIVER (board, bracket, transformer, and 31 relays) **$1128.08**
#86 - #06 (board, bracket, transformer, without relays) **$538.86**
#85 DPTO board **$348.85**
WINDRUSH EPROM PROGRAMMER **$375.00**
3" Binder ..... **12.00** 2" Binder **$9.00**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

DON'T SEE IT??? ASK! OUR BROCHURE HAS MORE COMPLETE DESCRIPTIONS AND SPECS. PHONE OR WRITE TODAY FOR YOUR COPY.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055